Date of acceptance Grade

Instructor

Two-stage sparse representation based abnormal crowd event detection in videos

Ege Can Özer

Helsinki May 6, 2020

Master's Thesis

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Ege Can Özer

Työn nimi — Arbetets titel — Title

Two-stage sparse representation based abnormal crowd event detection in videos

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master's Thesis	May 6, 2020	78 pages

Tiivistelmä — Referat — Abstract

Ubiquitous surveillance has become part of our lives to increase security and safety. Despite the wide application of surveillance systems, their efficiency is limited by human factors, such as boredom and fatigue; because most of the time, nothing unusual happens. In safety-critical applications, time is essential and it is vital to act fast to prevent costly incidents. This thesis proposes a two-stage abnormal crowd event detection framework based on k-means clustering in the first stage, and sparse representation based methods in the second stage, to alleviate the laborious task of video monitoring.

We conduct a literature review of 18 studies, where we specifically focus on sparse representation based methods. Accordingly, we choose the spatio-temporal gradient feature due to its simplicity, efficiency, and effectiveness in motion representation. After extracting features only from normal events, k-means clustering is applied to separate different motion feature clusters. Then, clusters with smaller samples, which are deemed to contain mostly abnormal features, are removed according to a threshold. In the second stage, we learn a dictionary for each remaining cluster using the approximate K-SVD algorithm. In testing, the reconstruction error of a feature against a learned dictionary and its sparse representation is used to determine an abnormality. We conduct extensive experiments on a standard dataset to evaluate the detection performance of the method. Furthermore, the effect of hyper-parameters in our method is investigated. We also compare our method with different methods to examine its effectiveness.

Results indicate that our abnormal event detection framework can successfully understand abnormal events in a scene while running in real-time at 161 frames per second. With a few exceptions, no significant advantage of the two-stage sparse representation approach over a single large dictionary was found. We speculate that these results may be influenced by a small sample size. Nevertheless, our approach, due to its unsupervised nature, can be adapted to different contexts without additional annotation effort and using only normal events from videos. Therefore it motivates us for further development.

ACM Computing Classification System (CCS):
Computing methodologies → Artificial intelligence → Computer vision → Computer vision tasks → Scene anomaly detection

Avainsanat — Nyckelord — Keywords
abnormal event detection, sparse representation, dictionary learning, video analysis

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Contents

1 Introduction

There is a growing demand for video surveillance due to increasing security and safety concerns. Now that closed-circuit television (CCTV) cameras are affordable and easy to deploy, they have become ubiquitous. The primary goal of video surveillance is to monitor continuously, identify suspicious events, and take further actions if possible, to prevent undesired circumstances. Despite vast numbers of CCTV cameras being used, their effectiveness is often limited by human factors. Some limitations are fatigue, boredom, and the capability of the operators to track a certain amount of cameras. Moreover, interesting events are infrequent so that constant human attention is required all the time. All these reasons necessitate an intelligent video analytics system to alleviate the labour-intensive task. In this work, we focus on understanding abnormal crowd events in videos.

Abnormal event detection aims to find out the behaviours that are considerably distinct from ordinary events in a time range and to locate a region of interests in a scene, such as by visualising to a user as bounding boxes, and possibly generate alarms or notify human agents to take action. For example, players running in a football match is a normal activity; whereas, a person suddenly running in the crowd may be inferred as an indication of illegal activity. What is abnormal depends on a context as well as application areas; still, we assume one can distinguish what constitutes an abnormal event in any scene.

Visual-based abnormal event detection covers various application areas; though, most applications are mainly for security and safety purposes. Some highlighted applications in surveys include detecting petty crime, violence, abandoned object, people loitering in an area, perimeter breach, intrusion, traffic law violation, and fall detection for older people [PW12, SRB12, LCW+15, MZ18]. This list is by no means exhaustive. It can be extended even more if we also include other research topics, such as action recognition.

Understanding "What is the abnormal event?" in a scene, is trivial for humans, whereas difficult for computers due to many challenges. The visual complexity of a scene is one factor: a sufficient quality of input data, lighting conditions, distance and position of sensors, the density of the moving objects in a scene. Understanding activities in a crowded scene is even a difficult task for a human due to occlusion. The techniques, interaction of objects, and contextual types of anomalies are studied differently in crowded scenes [PW12, LCW+15]. Another factor is the contextual

complexity of a scene. For instance, in the P-REACT project, it is noted that identifying bike thieves from the owners was very difficult and resulting in high false alarm rates [Doy16]. Humans may do the same tasks differently, or some tasks may exhibit a similar movement pattern. Other well-known challenges are concerning technical feasibilities and practical matters: high detection and low false alarm rate, runtime performance to meet a real-time surveillance need, budget requirements. Legal restrictions have started to be another variable in selecting video analytics after the General Data Protection Regulation (GDPR) law. All these factors in this topic make the abnormal event detection a non-trivial task.

Despite the challenges, vast amounts of research are available in the literature. Research efforts are mainly in two directions: feature extraction techniques and learning methods. Several feature extraction techniques exist for representation of events in a scene: handcrafted features such as spatio-temporal features [LSJ13, YLS17, GDL19], trajectory features [ZFFX11, CYL13, LLXZ18], or learned features, *e.g.* using deep learning [SFH16, SWXS18, CXYC19]. Earlier research put forward rule-based techniques; but, this paradigm has shifted towards data-driven methods due to limited robustness and scalability of handling unseen abnormal events [PW12].

In general, there is a great variety of data-driven methods for abnormal event detection, which are reviewed in great detail by the studies [SRB12, PW12, LCW$^+$15, MZ18]. However, as there are many different types of "abnormal", the use of traditional supervised learning methods in abnormality detection is impractical. The main reason is the annotated data requirements to learn every possible abnormal event. Therefore, most of the existing methods learn only normal events by employing unsupervised learning algorithms [ZFFX11, CYL13, LSJ13, SFH16, XLJ16, ISPA18]. In this thesis, we utilise sparsity-based methods, similarly to model normal events in a scene.

Sparse representation based methods have been proven to be useful for the abnormal event detection task. The main idea is to represent normal events as sparse linear combinations of a set of basis vectors that are learned from training videos. The key assumption is that normal events are likely to generate sparse reconstruction coefficients; whereas, abnormal events more dense representation. Results based on this assumption are mostly positive; many scientific studies that use the same idea have achieved state-of-the-art results, *e.g.* [ZFFX11, CYL13, LSJ13]. Although a limitation with these methods, in general, is the runtime performance, they perform well in detecting anomalies in the case of a limited amount of data in an unsupervised

fashion.

In this thesis, we propose a two-stage sparse representation based abnormal crowd event detection algorithm. First, we extract spatio-temporal volumes in which gradient along the temporal dimension is computed. Features are extracted from only normal events and do not require any annotation during the training stage. Hence, the learning task is unsupervised. Unlike some existing sparsity-based methods [LSJ13, MH17], we do not enforce in the training and testing stages for features to be from the same spatial location. Instead, after the feature extraction, we apply k-means clustering to find clusters representing different types of motion features. Next, we remove clusters that contain few samples according to a threshold. We assume these clusters contain noisy features. Our method in a way is similar to the method proposed by Ionescu *et al*. [ISPA18]. We learn a dictionary for each remaining cluster using the approximate K-SVD (AKSVD) algorithm [RZE08]. In the testing stage, for each learned dictionary, we compute the reconstruction error of a spatio-temporal gradient feature with respect to a learned dictionary and sparse coefficients. Based on a predefined normality threshold, and the minimum of reconstruction errors, an abnormal event feature is classified.

The rest of this thesis is organised as follows. Section 2 provides an introduction to the algorithms used in this thesis. In Section 3, we conduct a literature review on 18 studies, specifically focused on sparse representation based methods. Section 4 presents the proposed method. That is, a two-stage sparse representation based abnormal event detection framework. Next, experiments and results are demonstrated in Section 5. Section 6 is where we discuss the findings and limitations of our work. Lastly, Section 7 summarises and concludes the thesis.

2 Background

In this section, we introduce the preliminary concepts and algorithms that we use in our study. First, we formulate the abnormality detection problem within the sparse representation model.

Problem statement. From a given video, training features $X = [x_1, ..., x_N] \in \mathbb{R}^{P \times N}$ are extracted, where each $x_i \in \mathbb{R}^P$, P is a feature size and N is a sample size. A normal event dictionary $D = [d_1, ..., d_K] \in \mathbb{R}^{P \times K}$ is learned using X under some sparsity-inducing constraint ψ,where K is number of columns (also called *atoms* or *basis vectors*) in dictionary D. We wish to encode a new test instance x' as a *sparse* linear combinations of atoms $a \in \mathbb{R}^K$ in D. Mathematically this is expressed as

$$\begin{aligned} \hat{a} = &\arg\min_{a} \left\| x' - Da \right\|_2^2 \\ &\text{subject to} \quad \psi(a) \leq \zeta. \end{aligned} \tag{2.1}$$

The sparsity-inducing constraint $\psi(.)$ can be, *e.g.* ℓ_1-norm, and ζ is the sparsity controlling hyper-parameter. An abnormal event can be detected as a result of the large reconstruction error from $\|x' - D\,\hat{a}\|_2^2$ and using a predefined threshold.

One can see that there are two sub-problems to consider:

- How to encode input features sparsely from a fixed dictionary?
- Given a set of normal event features, how do we find a dictionary such that it promotes sparsity and represents input data well?

2.1 Sparse coding

Sparse coding (sometimes called sparse approximation) is a set of unsupervised algorithms to find an efficient and compact representation of an input signal $x \in R^P$ using a small number of elements from a set of basis vectors that are called atoms. Sparsity is motivated by the fact that the natural signals such as image, video, or audio contain high regularities; and they can be represented efficiently given that the objective is to maximise sparsity [OF96]. We formally express the sparse coding

problem as

$$a_* = \arg\min_a \|a\|_0 \quad \text{subject to} \quad x = Da. \tag{2.2}$$

In this formulation, a_* is the *sparse representation* of x, $a \in \mathbb{R}^K$ is the sparse coefficient vector, $\|.\|_0$ is the pseudo-norm function, often referred to as ℓ_0-norm, which counts non-zero entries of a vector, and $D = [d_1, ..., d_k] \in \mathbb{R}^{P \times K}$ is the dictionary. Each atom d_k is arranged as columns in D, and we assume D to be overcomplete, *i.e.* $P << K$. The term *sparse* means that a coefficient vector contains a small number of non-zero elements relative to the total number of atoms in a dictionary K. Figure 1 illustrates the schematic of sparse coding.

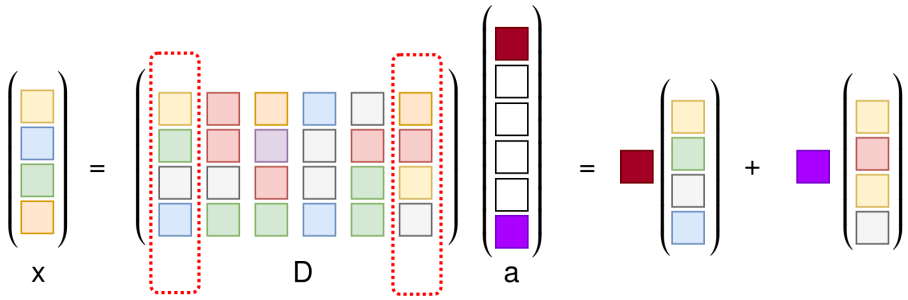

Figure 1: Schematic diagram of sparse coding

Actually, it is is not desired to compute the exact solution of (2.2) because of, *e.g.* the overfitting problem. Instead, an approximate solution $x \approx Da$ is used often:

$$\min_a \|a\|_0 \quad \text{subject to} \quad \|x - Da\|_2^2 \leq \epsilon. \tag{2.3}$$

In (2.3), an approximation of the sparse coding problem is shown using error constraint ϵ. There is also an equivalent formulation, where the sparsity is imposed as a constraint:

$$\min_a \|x - Da\|_2^2 \quad \text{subject to} \quad \|a\|_0 \leq \zeta. \tag{2.4}$$

Here, we note that we only consider the traditional sparse coding formulation, where input data is modelled using natural image *patches*. The patch based image models are used with great success in many image processing tasks, such as denoising, inpainting *etc.* in which an image is encoded patch by patch. Patches with a given size can be extracted from overlapping or non-overlapping regions from an image. An

extension to complete images can be achieved by, *e.g. convolutional sparse coding*; however, we will not study it in our work. We illustrate the idea of sparse coding for representing natural image patches in Figure 2. The figure shows a dictionary with 64 atoms learned using natural image patches, and a test sample patch is approximated as a sum of three dictionary atoms with the corresponding coefficients, *i.e. sparse linear combinations of atoms.*

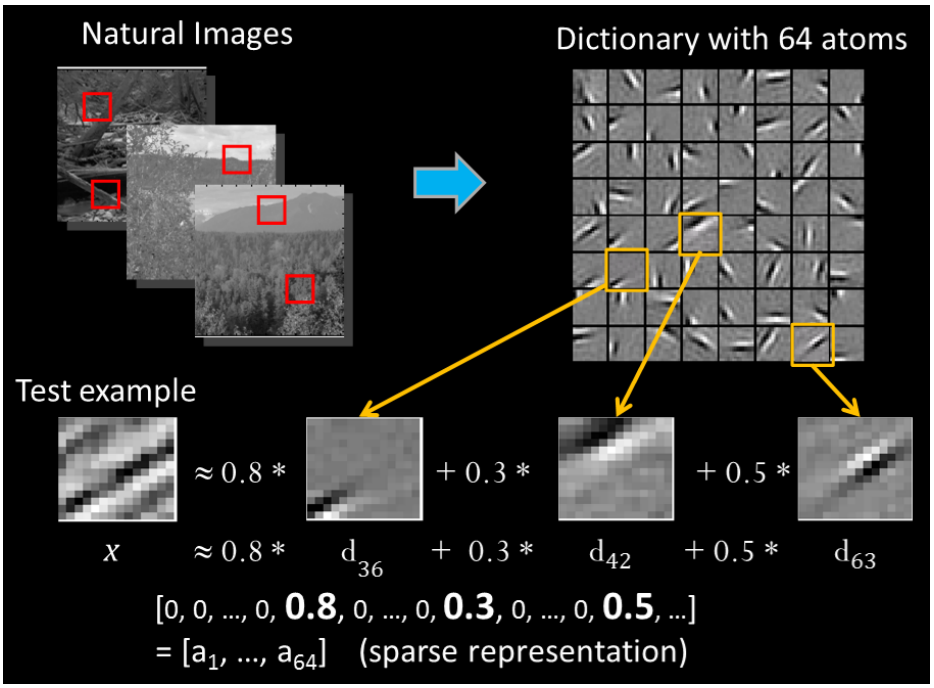

Figure 2: Sparse coding illustration on natural images. The image is based on the sparse coding lecture slides by Andrew Ng[2].

Solving the problem presented in (2.2) is known to be an NP-hard problem. One can observe this by simply solving it as a combinatorial optimisation problem. Suppose our goal is to represent an image patch using no more than $\zeta = 50$ atoms with a dictionary having $K = 1000$ atoms. The search space equals to $\binom{K}{\zeta} = \binom{1000}{50} \approx 9 \times 10^{84}$. Obviously, with more dictionary atoms, which is usually the case in image processing, the search space becomes even larger. Thus, the straightforward approach is infeasible.

In consideration of approximate solutions of (2.3) or (2.4), there are generally two strategies used: convex relaxations of ℓ_0-norm and *greedy* methods.

[2]Image classification using Sparse Coding by Andrew Ng - http://ufldl.stanford.edu/eccv10-tutorial/eccv10_tutorial_part2.ppt

ℓ_1-norm convex relaxation. First approach is convex relaxation of the ℓ_0-norm constraint in (2.2), which can be replaced by, *e.g.* ℓ_1-norm ($\|a\|_1 = \sum_i |a_i|$). We show the two common formulations of ℓ_1-norm approximations as

$$\min_a \|a\|_1 \quad \text{subject to} \quad \|x - Da\|_2^2 \leq \epsilon \tag{2.5}$$

and

$$\min_a \frac{1}{2} \|x - Da\|_2^2 + \lambda \|a\|_1 . \tag{2.6}$$

Note that the formulations in (2.5) and (2.6) are equivalent, and they fall into ℓ_1 regularised least square problems. The formula (2.6) is well-known in statistics and machine learning community as *least absolute shrinkage and selection operator* (in short *LASSO*) [Tib96]. Let $\lambda \in [0, \infty)$ denotes the regularisation penalty. For the larger values of λ, the sum of absolute values (ℓ_1-norm) of a is penalised, as a result we get sparse coefficients of a. In other words, the aim is to minimise ℓ_1-norm coefficients on contrary to directly seeking for the sparse representation. Over decades, there has been many efficient ℓ_1 solvers in the literature. Here we name only the few famous ones: LASSO [Tib96], *least-angle regression* (LARS) [EHJ$^+$04], *fast iterative shrinkage-thresholding algorithm* (FISTA) [BT09]. The main advantage of ℓ_1-norm approximation over greedy methods is the capability of inducing sparsity more effectively. Though the computational complexity is much higher than the greedy methods [MBP$^+$14].

Greedy methods. An alternative strategy to solve (2.2) is the use of greedy-type methods, which can be roughly categorised as *pursuit* or *thresholding* algorithms. Representative examples of pursuit algorithms are *matching pursuit* [MZ93] and *orthogonal matching pursuit* (OMP) [PRK93]. The basic working principle can be described in two steps: starting with an empty coefficient set, we add an "active" atom and optimise it iteratively. This idea is central to the matching pursuit algorithm [MZ93].

As opposed to pursuit algorithms, in thresholding algorithms after an atom selection, there is a pruning step. We show the *iterative hard thresholding* algorithm by Blumensath and Davies as an example thresholding algorithm [BD09]. In this method, the "hard thresholding" function does the pruning, which aims to find a fixed number of largest elements (controlled by a hyper-parameter) from input

data, and sets rest of elements to zero. By alternating between the two steps in each iteration, the method computes active atoms of the sparse coefficient vector a.

Orthogonal matching pursuit

The orthogonal matching pursuit algorithm, introduced by Pati *el al.* [PRK93], is a greedy method to approximate the solution of the sparse coding problems given by (2.3) or (2.4). The algorithm works almost identical for both of the variants, and only stopping conditions differ.

In each iteration, the algorithm first finds an atom that has the maximum correlation with the current residual, see line 5. Then, the algorithm computes the least square estimation error of the sub-dictionary of atoms that are found so far (line 7). Computing the least square $\|x - D_A a_{new}\|_2^2$ in a closed-form can be achieved by Moore-Penrose pseudoinverse $(D^T D)^{-1} D^T x$, where A is the active atom set, and D_A is the dictionary of chosen atoms in A. In another way, we can think this operation as an orthogonal projection of x onto D_A. The orthogonalisation ensures in each iteration that the residual is orthogonal to the current input approximation [RZE08]. It also implies that we only need to compute selected atoms once. Following then the residual is computed repeatedly until we reach the target sparsity ζ or the residual reaches to target error ϵ.

Algorithm 1 Orthogonal matching pursuit [PRK93]

1: **Input:** image patch $x \in \mathbb{R}^P$, dictionary $D \in \mathbb{R}^{P \times K}$, stopping criterion ζ or ϵ
2: **Output:** sparse coefficients vector $a \in \mathbb{R}^K$
3: **Initialise:** active atoms $\mathrm{A} = \varnothing$, residual $res = x$, $a = \vec{0}$
4: **while** $|A| < \zeta$ if ζ is given, or $\|x - Da\|_2^2 > \epsilon$ if ϵ is given **do**
5: $\quad a_{new} = \arg\max_a \frac{\left|d_a^T res\right|}{\|d_a\|_2}$ ▷ Find an atom with max. correlation with res
6: $\quad \mathrm{A} = \mathrm{A} \cup a_{new}$ ▷ Add the new atom to the set
7: $\quad a = \arg\min_{a_{new}} \|x - D_A a_{new}\|_2^2$ ▷ compute least square: $a = (D^T D)^{-1} D^T x$
8: $\quad res = x - D_A a$ ▷ Calculate the residual
9: **end while**
10: **return** a

However, we note that the inversion of $(D^T D)$ can be costly, especially inversion of a large dictionary. Instead of explicitly performing this operation, Cholesky or QR factorisation is performed [RZE08, MBP+14]. In our work, we used a very

efficient orthogonal matching pursuit variant called *batch-OMP* by Rubinstein *et al.* [RZE08], which employs Cholesky factorisation and performs sparse coding over a large set of inputs. Alternative acceleration techniques for the OMP algorithm are mentioned by Rubinstein *et al.* (2008) and Mairal *et al.* (2014) [RZE08, MBP+14].

2.2 Dictionary learning

In the previous section, we assumed that a dictionary for sparse coding is given as input. It was indeed the case in early research. A fixed set of atoms formed as columns in a dictionary is used to encode input signals sparsely. *Discrete cosine transform*, illustrated in Figure 3, is a famous example of a pre-fixed dictionary in compressed sensing literature, which is used in the JPEG image compression algorithm. Even though a predefined dictionary has a low computational cost, for more complicated tasks such as abnormality detection, it does not generalise well to data. It has been later shown by Olshausen and Field [OF97] that *learning* a dictionary from input data leads to a more efficient and compact sparse representation.

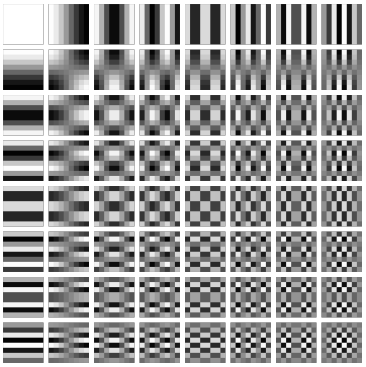

Figure 3: 8×8 two-dimensional discrete cosine transform frequencies are used as dictionary atoms. The image is taken from Wikipedia[4].

Let us formally define the dictionary learning problem. The goal of dictionary learning is to find a dictionary $D = [d_1, ..., d_K] \in \mathbb{R}^{P \times K}$ and a sparse coefficient matrix $A = [a_1, ..., a_N] \in \mathbb{R}^{K \times N}$ from the input data $X = [x_1, ..., x_N] \in \mathbb{R}^{P \times N}$ such that the reconstruction error $\|X - DA\|_F^2$ is minimised with a target sparsity constraint. It is formulated as the following optimisation problem

$$(D, A) = \arg\min_{D,A} \|X - DA\|_F^2 \quad \text{subject to} \quad \|a_i\|_0 \leq \zeta \quad \forall i, \tag{2.7}$$

[4]Wikipedia discrete cosine transform - https://en.wikipedia.org/wiki/Discrete_cosine_transform

where $\|.\|_F$ denotes the *Frobenius* norm, defined as $\|X\|_F = \sum_i^N \sum_j^P (X_{i,j}^2)$. Additionally, each dictionary atom d_k is required to satisfy $\|d_k\|_2^2 \leq 1$ constraint. This is to prevent a dictionary atom from being huge value, which in return it would make sparse coefficients very small. Such operation often is stated in the literature as the Euclidean projection Π_C onto a convex set C, where C is the convex set of matrices satisfying

$$C \stackrel{\text{def}}{=\!=} \{D \in \mathbb{R}^{P\times K}, \quad \text{subject to} \quad d_k d_k^T \leq 1 \quad \forall k = 1, ..., K\}. \tag{2.8}$$

The objective function for the optimisation problem presented in (2.7) is not jointly convex in dictionary D and sparse coefficient matrix A. However, it is convex if we alternate between optimising D while A is fixed, and vice versa. This is a very common technique in dictionary learning algorithms and it is called *alternating minimisation* or sometimes *block-coordinate descent*. We give an outline of the alternating minimisation scheme in Algorithm 2 for solving the dictionary learning problem (2.7).

Algorithm 2 Outline of alternating minimisation scheme to solve (2.7)

1: **Input:** input data $X \in \mathbb{R}^{P\times N}$, initial dictionary $D_0 \in \mathbb{R}^{P\times K}$ in C, target sparsity ζ, number of iterations T
2: **Output:** Learned dictionary D in C
3: **for** $t = 1,...,T$ **do**
4: Find sparse codes A with target sparsity ζ
5: Update dictionary D
6: **end for**
7: **Output:** Learned dictionary D, sparse coefficient matrix $A \in \mathbb{R}^{K\times N}$
8: **return** D

Having a good initial dictionary is important as dictionary learning algorithms often aims to improve a given initial dictionary rather than finding an optimal one [AEB06]. There are a few choices used in practice to compute an initial dictionary:

1. Initialise a dictionary using random values.
2. Take random samples from input data.
3. Initialise dictionary using clustering algorithms, *e.g.* the K-means algorithm can be used to take cluster centres as dictionary atoms.

4. Use a matrix factorisation technique, *e.g.* the singular value decomposition to compute an initial dictionary.

Different choices of sparse coding algorithms, dictionary update steps, and prior assumptions about input data have led to many variants of dictionary learning algorithms. Here, we have only covered unsupervised dictionary learning algorithms. The most prominent dictionary learning algorithms include the stochastic projected gradient descent [OF97], the method of optimal directions [EAH99], the K-SVD [AEB06, RZE08], and the online dictionary learning [MBPS09]. We note that all these algorithms are following the similar alternating minimisation technique. Now we will review these algorithms in detail.

Stochastic projected gradient descent. Olshausen and Field [OF97] propose the first algorithm to solve the dictionary learning problem presented in (2.7). However, they model the sparsity as a prior probability distribution over sparse codes rather than, *e.g.* ℓ_1-norm. For simplicity, we show this as $\psi(a)$ as in a sparsity-inducer function. In each iteration, the algorithm takes only one sample from the input. It finds sparse codes and proceeds to dictionary update. The problem considered now is the following

$$\min_{D} \sum_{i=1}^{N} f(x_i, D) \tag{2.9}$$

and

$$f(x_i, D) \stackrel{\text{def}}{=\!=} \min_{a} \|x_i - Da\|_2^2 \text{ subject to } \quad \psi(a) \leq \zeta. \tag{2.10}$$

In the dictionary update step, gradient descent of (2.10) with respect to dictionary D is calculated, *i.e.* $\nabla_D f(x_i, D)$. The gradient is given by

$$\nabla_D f(x_i, D) = 2(Da_i - x_i)a_i^T. \tag{2.11}$$

After, the dictionary is updated, it is then projected onto unit ℓ_2 ball ($d_k = \frac{d_k}{\|d_k\|_2}$) to prevent dictionary atoms from growing out-bound. This is shown as

$$D_{t+1} = \Pi_C(D_t - \eta_t \nabla_D f(x_i, D_t)), \tag{2.12}$$

where η is the step size in gradient descent. Stochastic gradient descent performs well if a good η is found in data-independent way [MBPS09]. However, in practice, it is very challenging to find an appropriate value for η. We give a summary of the stochastic projected gradient descent in Algorithm 3.

Algorithm 3 Stochastic projected gradient descent

1: **Input:** input data $X \in \mathbb{R}^{P\times N}$, initial dictionary $D_0 \in \mathbb{R}^{P\times K}$ in C, target sparsity ζ, number of iterations T, step size η
2: **Output:** Learned dictionary D in C
3: **for** t=1,...,T **do**
4: Take one sample x_i from X randomly
5: Find sparse code a_i with target sparsity ζ:

$$a_i = \arg\min_a \|x_i - Da\|_2^2 \text{ subject to } \quad \psi(a) \leq \zeta \tag{2.10}$$

6: Update dictionary D:

$$D_{t+1} = \Pi_C \left(D_t - \eta_t \, 2(Da_i - x_i)\, a_i^T\right)$$

7: **end for**
8: **Output:** Learned dictionary D, sparse coefficient matrix $A \in \mathbb{R}^{K\times N}$
9: **return** D

Instead of taking one sample in each iteration, a practical extension is to take a set of samples from input data and perform the same from line 4–6. The extension is called *mini-batch* gradient descent in the literature. We do not show this variant here.

Method of optimal directions. The algorithm proposed by Engan *et al.* is addressing the formulation (2.7) [EAH99]. They have applied the method of optimal directions (MOD) algorithm to approximate speech and electrocardiogram signals. Similarly, the algorithm alternates between sparse coding and dictionary update steps, as in Algorithm 2. Sparse coding is performed individually to each input sample using any methods presented in the earlier section. The main difference is in the dictionary update step. With a fixed sparse representation matrix A, the dictionary is updated iteratively from the analytical solution of $D = \arg\min_D \|X - DA\|_F^2$. This can be found as $D = X(A)^+$, where $(A)^+$ denotes a Moore-Penrose pseudo-inverse, so that the dictionary is updated according to $D = XA^T(AA^T)^{-1}$ [EAH99].

K-SVD. A more recent algorithm to efficiently learn overcomplete dictionaries for sparse representation is proposed by Aharon *et al.* [AEB06]. The K-SVD algorithm similarly iterates between sparse coding and dictionary update steps. Since it is addressing (2.7) to find sparse coefficients, any pursuit algorithms can be used. However, due to the efficiency reasons, often, the OMP algorithm is used. Differently from the MOD, instead of matrix inversion, a dictionary update is performed one atom at a time in a block-coordinate fashion. That is, optimisation is done for each dictionary atom individually while keeping other variables fixed.

The main contribution of the K-SVD algorithm is in the dictionary update step. When updating the dictionary atom d_k, for each $k = 1, ..., K$, the algorithm first seeks to find an index set from the input data X, in which the dictionary atom d_k is used in the sparse representation. Let us denote the index set of these data points as I. Next, for all indices in I, except the kth atom, *i.e.* without the contribution of d_k, the residual matrix E is computed as $E = X_I - D\,A_I$. The K-SVD algorithm updates the dictionary atom d and sparse coefficients a by minimising

$$(d, a) = \arg\min_{d,a} \left\| E - d\,a^T \right\|_F^2 \quad \text{subject to} \quad \|d\|_2 = 1. \tag{2.13}$$

The K-SVD algorithm uses the singular value decomposition (SVD) technique to factorise the residual matrix $E = U\Sigma V^T$, and d_k is updated according to the first column of U, *i.e.* $d_k = U_1$. The dictionary is updated until all atoms are done [AEB06, RZE08].

However, the SVD is an expensive operation as well as memory consuming in the K-SVD algorithm, due to the size of E being relative to the input X. And in practice, the input can be huge. Rubinstein *et al.* propose the *approximate K-SVD* algorithm (AKSVD) [RZE08] to address this. The residual matrix E computation is alternated to another iteration to speed up, in which the dictionary atom d and sparse coefficients a are given by

$$\begin{aligned} d &= \frac{Ea}{\|Ea\|_2}, \\ a &= E^T d. \end{aligned} \tag{2.14}$$

In fact, Rubinstein *et al.* show that a single iteration gives a close result compared to the full computation via the SVD [RZE08]. In our experiments, we have used the approximate K-SVD to learn normal event dictionaries. We show the approximate

K-SVD in Algorithm 4.

Algorithm 4 Approximate K-SVD [RZE08]

Input: input data $X \in \mathbb{R}^{P \times N}$, initial dictionary $D_0 \in \mathbb{R}^{P \times K}$ in C, target sparsity ζ, number of iterations T
Output: Learned dictionary D in C
for $t = 1,...,T$ **do**
Find sparse codes A using any pursuit algorithm *e.g.* the OMP algorithm:
  for $n = 1,...,N$ **do**
    $$a_n = \arg\min_a \|x_n - Da\|_2^2 \text{ subject to } \quad \|a\|_0 \leq \zeta \tag{2.4}$$
  end for
Update dictionary D:
  for $k = 1,...,K$ **do**
    $D_k = 0$ ▷ No contribution of the kth atom
    $I = \{\ i \in \{1, ..., N\} \mid a_{k,i} \neq 0\ \}$
    $a = A_{k,I}^T$
    $d = X_I a - DA_I a$ ▷ Update the dictionary atom d, Eq. (2.14)
    $d = d/\|d\|_2$
    $a = X_I^T d - (DA_I)^T d$ ▷ Update the sparse coefficients a, Eq. (2.14)
    $D_k = d$ ▷ Set the updated atom to the column in D
    $A_{k,I} = a^T$ ▷ Update the kth row and the columns in I
  end for
end for
return D

Online dictionary learning. So far, we have seen the dictionary learning algorithms which require the entire input data before the learning stage. In the literature, this is called offline learning. Offline learning may be problematic in certain situations: (1) training over very large input data is computationally infeasible due to resource constraints, *e.g.* memory. (2) data may not be stationary so that "re-training" becomes necessary. Online machine learning algorithms are a common technique to address this.

In the same vein, Mairal *et al.* [MBPS09] propose an *online dictionary learning* algorithm that updates the dictionary iteratively as new data become available. It

helps to reduce memory requirements to learn a dictionary from large input data, and the dictionary can be adapted to new data dynamically.

The online dictionary learning algorithm [MBPS09] employs the same alternating-minimisation scheme as in Algorithm 2. However, this time, sparse coding is computed using ℓ_1 convex relaxation. Particularly, Mairal *et al.* [MBPS09] suggest that using the least-angle regression (LARS) algorithm [EHJ+04] in dictionary learning leads to a more robust result. Then, the algorithm updates a dictionary iteratively using the block-coordinate descent approach [MBPS09]. The main advantage of the block-coordinate descent approach is that it is hyper-parameter free and effective in practice.

2.3 K-means clustering

K-means is one of the most famous and widely used clustering algorithms in data analysis. The standard k-means algorithm was first proposed by Stuart Lloyd in 1957 as a technique for pulse-code modulation, and later on, published in 1982 [Llo82]. Therefore, the k-means algorithm is often called Lloyd's algorithm in the literature. Clustering is a common approach in the unsupervised learning methods for abnormality detection [SRB12]. However, we use it only after the feature extraction step to cluster different motion features and remove small clusters, which are deemed to contain noisy features.

Let us first define the k-means formulation in a generic clustering problem. The k-means algorithm requires two input parameters: a set $X = \{x_1, ..., x_n\}$ of n data points and an integer k denoting the number of desired clusters. The goal is to find k centres $C = \{c_1, ..., c_k\}$, and assign each data point x into k sets $S = \{s_1, ..., s_k\}$ such that the sum of squared distances for every data point and its distance to the closest centre are minimised. This can be shown formally as the following optimisation:

$$\min_{S} \sum_{i=1}^{k} \sum_{x \in s_i} \|x - c_i\|^2. \tag{2.15}$$

The global optimum solution of (2.15) is computationally difficult problem. The k-means or Lloyd's algorithm (and many of the variants) converges to a local optimum solution and refines in an iterative way. The pseudo-code for the k-means algorithm is given in Algorithm 5.

Selecting proper initial cluster centres is one important factor that affects the per-

Algorithm 5 K-means algorithm [Llo82]

1: **Input:** Set $X = \{x_1, ..., x_n\}$ of n data points, the number of desired clusters k
2: **Output:** Cluster centers $C = \{c_1, ..., c_k\}$, Set of clusters $S = \{s_1, ..., s_k\}$
3: Arbitrarily select k inital cluster centers $C = \{c_1, c_2, ..., c_k\}$
4: **do**
5: $\forall i \in \{1, ..., k\}$, assign each data point x to its closest cluster center c_i
6: $\forall i \in \{1, ..., k\}$, update the cluster centers c_i as

$$c_i = \frac{\sum_{x \in s_i} x}{\textbf{cardinality}(s_i)}.$$

7: **while** C change ▷ Or until reaching the maximum iteration
8: **return** C, S

formance of the k-means algorithm. For instance, assuming data has well-separated clusters, selecting closer data points as cluster centres would produce bad clusters.

Instead of sampling centres arbitrarily, Arthur and Vassilvitskii [AV07] propose a better initialisation heuristic for the k-means algorithm, which they have called k-means++. The initialisation heuristic calls for scattered cluster centres. This is achieved in the following. First, an initial centre c_1 is arbitrarily selected from the data. Let $\text{Dist}(x, C)$ denote the distance from a data point x to the closest cluster centre in C. The next cluster centre $c_i = \hat{x} \in X$ is selected with a probability given by

$$\frac{\text{Dist}(\hat{x}, C)^2}{\sum_{i=1}^{n} \text{Dist}(x_i, C)^2}, \tag{2.16}$$

and this process is repeated until all k initial cluster centres are selected. In other words, the probability of choosing $\hat{x}$ as a new cluster centre is high if $\hat{x}$ is not close to the already chosen cluster centres. After we choose all initial centres using k-means++, clustering is identically performed to the k-means algorithm [AV07].

3 Literature review

In this section, we review the related studies in abnormal event detection research. Because there is a vast literature in this area, first we describe the methodological approach we use for this section. Next, a total of 18 papers are reviewed in three separate sub-sections: (1) representation of the events, (2) classification frameworks, (3) evaluation methodology and standard datasets. At the end of the section we express our findings resulted from the literature review.

3.1 Methodological approach of the literature review

Overview. First, we look through the topic at hand to understand the scope of the literature. In order to cover a sufficient amount of publications, we have used Scopus, Elsevier's peer-reviewed literature database. In the design of the query, we have considered having enough coverage with as few outliers as possible. The literature and application areas are vast; as a result, the glossary used among the researchers differs. Hence, we have included nearby terms to the search query, such as anomaly detection, unusual behaviour, suspicious event recognition, and so on. We have explicitly left out some related computer vision topics (object detection, activity recognition) to narrow down the scope. Had we included these topics within our search query; it would have been resulted in more papers to inspect. The Scopus query, as given in Figure 4, has yielded 505 studies in the literature.

In this topic, we see an increasing trend each year with few exceptional years and peak at 54 papers in 2018. The average number of documents published each year is 20. The distribution of abnormal event detection researches over the years is illustrated in Figure 5. Majority of the studies are conference papers, which is in general similar in computer vision research. Top three publishers by country are China, the United States, and India. Next, we examine the related surveys to see common findings and missing factors in abnormal event detection research.

Related surveys. Existing surveys in this topic cover broader problem formulations, assumptions, various applications, and a variety of techniques of event description and classification models [PW12, SRB12, LCW+15, MZ18]. First considering the environment, according to the findings in these surveys, the individuals are the most common abnormality target, following the traffic, and crowds. The lack of anomaly detection on inanimate objects is noted in [SRB12]; though, some ap-

```
TITLE-ABS((abnormal OR anomaly OR unusual OR suspicious)
          (detection OR recognition)
          (event OR behaviour OR scene OR crowd OR surveillance)
          (image OR camera OR video))␣
AND SUBJAREA(comp)␣
AND KEY(("pattern recognition") OR ("computer vision") OR
        ("video surveillance") OR ("abnormal event detections"))␣
AND NOT TITLE(object OR activity OR action OR tamper OR intrusion)␣
AND NOT KEY(network OR iot)␣
AND NOT ALL(medical OR medicine)␣
AND (LIMIT-TO(LANGUAGE,"English"))
```

Figure 4: The Scopus query is used to collect the relevant studies in the abnormal event detection research area. The search query has resulted in 505 documents on 2019-12-31, which includes a wide range of techniques.

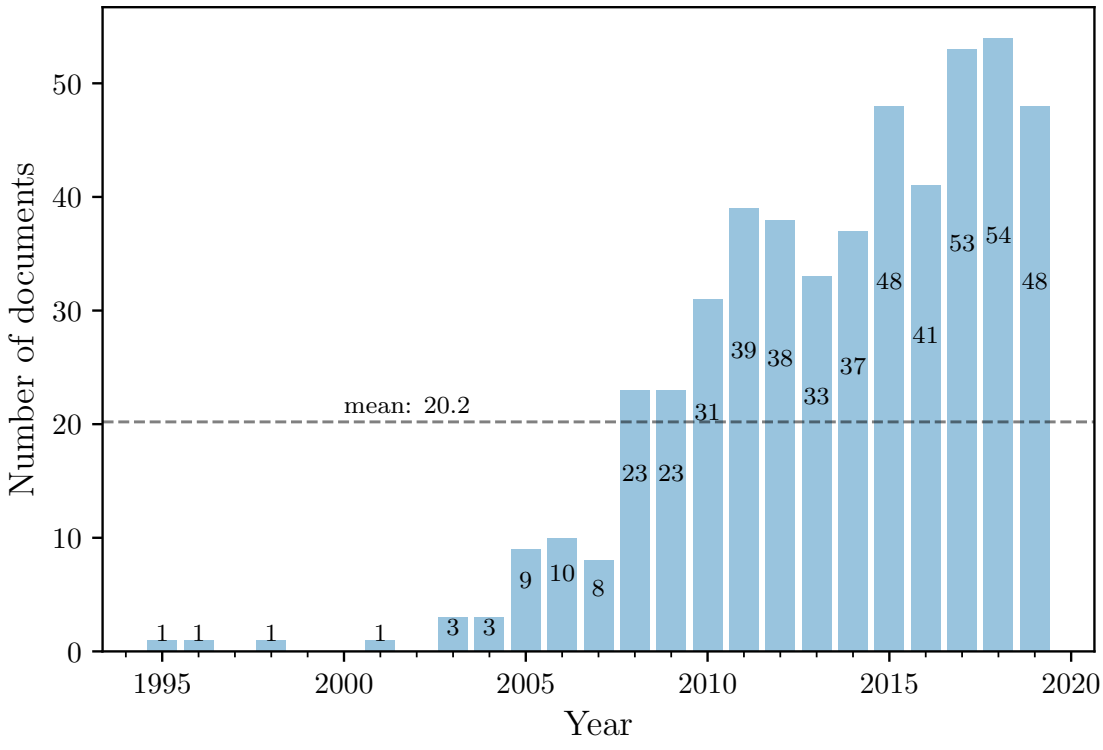

Figure 5: The distribution of abnormal event detection studies published between 1995 and 2019. The Scopus query in Figure 4 is used to collect the documents.

plications regarding the security of an object, for example, to prevent shoplifting, commercially exist. Majority of the researches apply anomaly detection to video data that is collected from the visible-spectrum camera. Other sensors, such as infrared camera or audio, are rarely used. Sodemann *et al.* [SRB12] mention most of the research addresses higher resolution images and considers fields of view that cover only the middle range (10-100 m^2). Extreme cases, such as very near or far, are less likely studied.

Comparable to the other computer vision tasks, we observe the great variety of event representation techniques and learning models applied for abnormal event detection. A recent survey by Mabrouk and Zagrouba [MZ18] list popular feature types used for abnormal event detection: motion-based information, interest points, spatio-temporal volumes, shape features, texture features, object tracking. The abnormal event detection algorithms are generally categorised as supervised, unsupervised, or semi-supervised. Most frequently, normal events are learned. However, despite the popularity, none of the surveys mention deep learning-based methods either used at the event representation or learning stage. Existing video surveillance systems are only pointed out by Mabrouk and Zagrouba [MZ18], but their effectiveness is not discussed. Earlier surveys [PW12, SRB12] state the lack of available datasets for evaluating the effectiveness of the algorithms. However, we note that it is no longer valid. In the following sections, we will present the commonly used datasets. Computational requirements and run time performances of the algorithms are often not included in the reviews.

Due to the visual difficulties and distinct semantics, the techniques used for crowded scene analysis and the interaction among the objects involved in a scene (individual, group) are treated separately. In this regard, Li *et al.* [LCW+15] examine the techniques that are used in different video tasks, such as motion pattern learning, behaviour analysis, and anomaly detection; but only from the crowded scene's point of view.

Conclusion. Abnormal event detection is a widely studied topic, and there are several ways to categorise research. Popoola and Wang give a sample of possible themes for grouping research publications, such as unsupervised event modelling, hybrid of generative and discriminative models [PW12]. Differently from the other surveys, we seek to create a more focused review on sparse representation based methods, either used at event representation or learning stage. So we refine the

search query and manually select the relevant publications. We have ended up reviewing 18 papers in this section.

3.2 Event representation

A high-resolution image consists of millions of pixel data. For a video, data size is much larger. With a more compact representation of data, it is more efficient to perform computer vision algorithms. As a result, scientists have long attempted to find various feature extraction methods to find the relevant piece of information for a particular task in image processing and computer vision. Such abstraction not only gives data compactness but also makes features robust to a variety of changes in illumination, rotation, scale, and so on. Here, we check the commonly used features in sparse representation based methods for the abnormal event detection task. Table 1 shows a summary of features types and representation methods that are used in these papers.

	Feature types and representation	References
Hand-crafted	Spatio-temporal volumes	[LSJ13, SFH16, YLS17, MH17, LSWJ18, GDL19]
	Optical-flow based features	[ZFFX11, CYL13, HFWW13, YLZ+16, HSS18, YFL18]
	Appearance features	[ZFFX11, CWZ+17, YFL18]
	Interest points	[ZFFX11, DBR16, YFL18]
	Trajectory based features	[CWZ+17, LLXZ18]
Learned	Autoencoder	[SFH16, SWXS18, LSWJ18]
	Convolutional neural networks	[CXYC19]
	Sparse dictionary learning	[XLJ16]

Table 1: Common feature extraction and representation methods, which are used for abnormal event detection.

Assuming in any context, one can find what constitutes an abnormal event in a scene; features should be highly descriptive to represent both local or global abnormal events, which may exist spatially, temporally, or both as mentioned in [PW12, CYL13]. Local abnormality means that an activity of an individual is considered significantly different from its vicinity; whereas, in global abnormality, the global activity in a scene that changes abruptly can be considered abnormal. For instance, a person walking towards in the wrong direction in a subway entrance is a

local, and a sudden crowd movement due to an explosion is a global abnormal event. In any case, features play an essential role to represent events in a scene, whether it is to detect a local or global abnormal event.

Spatio-temporal volumes have been found useful in motion representation. Within each predefined local region in an image, they are sampled temporally. An illustration of spatio-temporal volumes is shown in Figure 6. One major disadvantage is the large feature space; as a result, immense memory requirements. Once spatio-temporal volumes are extracted, image properties, such as texture, appearance, motion features can be described by applying various algorithms.

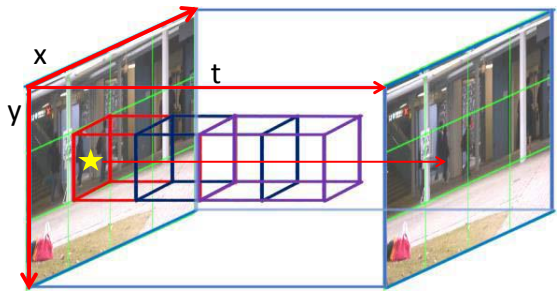

Figure 6: Spatio-temporal volumes are sampled along time axis in a video, the figure is based on Yu *et al.* [YLS17]. The yellow star illustrates a hypothetical interest point, where spatio-temporal volumes are extracted around this point.

One common approach is to use the 3D gradient features to represent both appearance and motion information in a scene. We note that the 3D gradients that are applied to spatio-temporal volumes are used commonly due to its descriptiveness and low computational requirements by several studies [LSJ13, YLS17, MH17, LSWJ18, GDL19]. When the feature is applied along the temporal dimension of a video, it is sometimes referred to as the spatio-temporal gradient feature. A common setup is to extract spatio-temporal volumes from non-overlapping local regions and apply the 3D gradients afterwards. Two studies use the feature in different variations to improve the classification result. Masoudirad and Hadadnia [MH17] extract the features from near and far planes in a scene, and train them separately. Geng *et al.* remove background in a scene using a Gaussian mixture model before the feature extraction step.

Instead of extracting every local feature from predefined regions, interest points can be used to find out regions based on mathematical formulations. An advantage is that the feature space is sparser than the pixel representations. Further, interest points are robust to certain changes in images, such as rotational invariance. Corner

detection is a well-known example of interest points. In the video domain, interest points are often extended to the spatio-temporal domain. Several studies use interest points [ZFFX11, DBR16, YFL18]. In Figure 6, we illustrate a spatio-temporal volume extracted from a hypothetical interest point marked as a yellow star.

The optical flow feature consists of motion direction and magnitude components. They are considered as global features because the computation involves the entire image. It is computationally expensive but expressive in motion representation. Cong *et al*. [CYL13] propose the *multi-scale histogram of optical flows (MHOF)* and a set of spatio-temporal basis to represent global and local abnormality. The MHOF feature is used by several studies [HFWW13, YLZ+16, HSS18].

Optical flow features are also used in combination with other features in different studies. For example, Zhao *et al*. [ZFFX11] extract spatio-temporal interest points from sliding windows; followed by computing the *histogram of oriented gradient (HOG)* and the *histograms of optical flow (HOF)* features representing the final feature vector. Yuan *et al*. [YFL18] follow the same scheme; they instead compute spatio-temporal interest points from non-overlapping regions. The difference is in the application of sliding windows operation. Each sliding window also adds neighbouring patches by a given step size, and therefore one obtains more features.

Some studies model motion trajectories in a scene to learn abnormal. Lv *et al*. [LLXZ18] propose the *repulsive forces* feature for crowd anomaly detection, where tracklets are extracted by a deep optical flow model. Chen *et al*. [CWZ+17] use the HOG feature followed by the Kalman filter to construct motion trajectories; they use this to learn vehicle behaviour.

Until now, we have seen previous research that employs hand-crafted features. Recently, learned features, such as artificial neural networks have received attention due to superior performance over hand-crafted features in certain computer vision tasks, *e.g*. object detection. In [CXYC19], a deep 3D convolutional neural network model (C3D) is used to extract features. In [SFH16, SWXS18, LSWJ18] the latent information in a scene is constructed by autoencoder models in which normal events are learned in an unsupervised way. Xue *et al*. [XLJ16] use the *histograms of sparse codes (HSC)*, such features are learned from data by employing sparse dictionary learning. They have shown that the HSC is more descriptive than the combination of HOG-HOF features. Usually, these methods require a large amount of data and high processing power to find informative features.

3.3 Classification frameworks for abnormal event detection

Methods based on sparse representation are effective for the abnormal event detection task. The main idea is to represent input data as sparse linear combinations of a set of atoms, typically from a learned dictionary. Results show that normal events could be reconstructed using a small number of atoms with a small reconstruction error; whereas abnormal events need a large number of atoms resulting in a high reconstruction error. Here we present briefly a variety of classification frameworks for abnormal event detection that primarily use sparse representation.

Starting from the earlier studies, Zhao *et al*. [ZFFX11] propose unsupervised abnormal event detection based on online dictionary learning algorithm for sparse coding introduced by [MBPS09]. Cong *et al*. [CYL13] introduce *sparse reconstruction cost* criterion to detect abnormal event. However, their approach is offline (batch) and suffers from the concept drift problem. Han *et al*. [HFWW13] address this issue, and they propose an adaptive online dictionary learning method. We observe that these methods are limited by particularly low time efficiency in testing.

Two studies consider the geometric structure of the training data to maintain local properties of data points as well as the correlation to its neighbours [YLZ$^+$16, YFL18]. In the method by Yi *et al*. [YLZ$^+$16], each sample's dictionary atoms are linearly reconstructed by nearest k neighbours, they name it *constrained sparse representation* model, and for testing *sparse reconstruction cost* is adopted [CYL13]. On the other hand, Yuan *et al*. [YFL18] employ *reference events*, which are learned from a Gaussian mixture model, to enforce structural information. They use smoothness regularisation jointly formulated within the sparse representation framework algorithm. We see that considering the manifold structure of input data improves the detection accuracy in these papers compared to, *e.g*. [CYL13].

In video processing, representing the input using linear combinations of atoms based on a single large dictionary may cause some complications, most critically the high computation cost. Lu *et al*. [LSJ13] address this issue and propose a very efficient method that can operate in real-time with a decent detection rate, which they call *sparse combination learning* framework. Sparsity constraint is coded directly as a set of combinations (some studies use *group dictionary* term instead) of bases, where each combination contains a fixed number of dictionary atoms. With this change, a large scale dictionary problem is relaxed into many small-scale combinations. During testing, the goal is to find a combination with the minimum reconstruction error. Results are mostly positive and motivating others to adopt this method. Two studies

use *sparse combination learning* framework directly, they only differ in the feature description step [MH17, GDL19]. Three studies [YLS17, LLXZ18, LSWJ18] contribute in the following way: (1) Yu *et al*. use low-rank approximation and sparse combination learning to capture normal event dictionaries [YLS17]. Advantages of the method are: the number of dictionary atoms is adaptively found, improved abnormal detection rate, and reduced training time. (2) Lu *et al*. propose an online variant of their previous algorithm to handle concept drift problem and high memory requirements [LSWJ18]. The online variant achieves competitive result in detection compared to the former method [LSJ13], while adapting to the changes in the scene. (3) An alternative dictionary updating strategy is proposed by Lv *et al*. [LLXZ18]. After the initial group dictionaries learning, the local or global update happens according to the reconstruction error of two consecutive group dictionaries.

Dutta *et al*. propose a new definition for an outlier data point: "... a data point is deemed an outlier if it is strongly constituted of dictionary atoms that rarely constitute other points." [DBR16]. The rarity of dictionary atoms is calculated using negative log activity ratio (NLAR). Then, they propose rarity based outlier detection (RODS) algorithm with both online and offline variants applied in sparse coding, and utilise in three different computer vision tasks: saliency detection, abnormal event detection, change detection. They achieve better abnormal event localisation compared to, *e.g*. [CYL13].

The study by Xue et al. formulate abnormal crowd event detection in optimal path discovery problem solved by Max-Path algorithm [XLJ16]. Methods here so far have only considered normal events in training. Interestingly, He et al. use both normal and abnormal events given as training data for abnormal detection based on *multiple instance learning (MIL)* and dictionary learning [HSS18]. They use a MIL classifier to determine only the positive bags (abnormal), and the reconstruction error from a dictionary gives the final result. In [CWZ+17], sparse representation and similarity based trajectory classifiers are used to learn vehicle behaviours. While the trajectory classifier alone has a competitive result, learning the sparse representation helps to improve the performance. In the comparative analysis, the method has achieved the best result among the other algorithms.

Furthermore, three studies utilise artificial neural network models, particularly autoencoders and convolutional neural networks [SFH16, SWXS18, CXYC19]. Sabokrou *et al*. utilise a *sparse autoencoder* model to compute the sparsity and a standard autoencoder to calculate the reconstruction error of normal patches [SFH16]. Ab-

normal events are classified accordingly to the cascade classifier. On the other hand, Sun *et al.* [SWXS18] introduce the first formulation of sparse representation framework based on *variational autoencoders*. Chu *et al.* use the C3D model to extract features, and sparsely encodes to learn new features [CXYC19]. After learning the features, they utilise the *sparse combination learning* framework for abnormal event detection [LSJ13]. Although the cost of their complexity, hybrid approaches are indeed effective in moving toward more robust anomaly detection.

3.4 Evaluation methodology and standard datasets

Evaluation methodology. There are mainly four evaluation levels applied to assess the predictions from an anomaly detection system. We use the definitions from several studies [MLBV10, CYL13, LSWJ18], and present each evaluation level in the following.

Frame-level: A prediction is flagged as abnormal if there is at least one abnormal feature (instead of "feature" sometimes "pixel" term is used) in a frame. Predictions from each frame are compared against frame-level ground truths. Datasets without pixel annotations can be used, as this evaluation is not interested in evaluating the locations of abnormal events. This is the most frequently applied evaluation level.

Object-level: A prediction is flagged according to spatial locations of anomalies. To evaluate a prediction, intersection over union is computed as:

$$\frac{A_p \cap A_g}{A_p \cup A_g} \geq \tau_o, \tag{3.1}$$

where A_p is the predicted abnormal pixels, A_g is the ground truth regions, and τ_o is the object-level threshold. An advantage is that datasets do not have to provide pixel-perfect annotations to evaluate the predicted localisation. We see that this measurement is used in two studies [YFL18, LSWJ18].

Pixel-level: Similarly to the object-level evaluation, a prediction is flagged according to the locations of anomalies. The locations of the predicted abnormal pixels that are compared to the ground truth abnormal masks must be equal or higher than 40%, which is calculated similarly to equation (3.1).

Event-level: In this evaluation, the motivation is to assess the method's capability in detecting abnormal events in a more general way. Predictions about localisation and frame are extrapolated to event level predictions. According to the Cong *et al.*, if more than one frames are detected as abnormal (false-negative frame predictions

can also be tolerated), and the location of an anomaly in a scene is determined correctly then an event-level prediction is flagged [CYL13]. However, the authors do not provide sufficient information on how systematically an event-level prediction should be detected.

Regardless of the evaluation levels, performance metrics derived from the binary classification test are used. First, we present the fundamental terminologies in binary classification in Table 2. In the table, the columns indicate prediction classes, rows indicate ground truth classes, and indexes of the 2×2 matrix are the results of a prediction.

		Prediction	
		Normal	Abnormal
Ground truth	Normal	True negative (TN): How many normal events are truly normal events?	False positive (FP): Prediction causes a false alarm.
	Abnormal	False negative (FN): A true abnormal event is missed.	True positive (TP): An event predicted as abnormal and it is truly an abnormal event.

Table 2: Binary confusion matrix for abnormal event detection problems.

Using the results from a binary confusion matrix, one can calculate the standard performance metrics to evaluate a model. Table 3 introduces the definitions of the performance metrics, which are commonly used in the reviewed studies. We note that primarily, metrics that are derived from the *receiver operating characteristic* (ROC) curve are used to evaluate the methods. The ROC curve is used to plot the performance of a classification model at multiple threshold settings, and consists of two parameters: true positive rate (TPR) compared against false positive rate (FPR). Higher the AUC and EDR better a model performs. Lower the EER better a model performs. The ROC curve of a good model is closer to the axis on the top-left side of a figure. Moreover, the accuracy, F_1, TPR (also known as recall), and precision are also frequently used metrics, and a higher value of these metrics indicate good performance.

For runtime performance analysis *frames per second* (FPS) is a commonly accepted metric. Runtime performance analysis is usually done at feature extraction, model prediction, or for overall method speed.

Metric	Definition
True positive rate (TPR)	$\frac{\text{TP}}{\text{TP+FN}}$
False negative rate (FNR)	$\frac{\text{FN}}{\text{FN+TP}} = 1 - \text{TPR}$
False positive rate (FPR)	$\frac{\text{FP}}{\text{FP+TN}}$
Precision (P)	$\frac{\text{TP}}{\text{TP+FP}}$
Accuracy	$\frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$
F_1 score	$\frac{2 \cdot \text{P} \cdot \text{TPR}}{\text{P+TPR}}$
Area under curve (AUC)	Area under the ROC curve
Equal error rate (EER)	The common value at which FPR = FNR
Equal detection rate (EDR)	EDR = 1 - EER

Table 3: Commonly used performance metrics for anomaly detection methods.

Standard datasets. Secondly, we note the availability of various abnormal event detection datasets in recent years. In this study, we are specifically interested in understanding abnormal crowd events. Therefore, we present the most frequently used datasets, which are designed for understanding crowd events. We do not mention rarely used datasets, or the datasets that are used in different contexts, *e.g.* vehicle behaviour learning.

Upon checking the publications in our review, we see that there are four commonly used datasets for abnormal crowd event detection. Among them, the UCSD dataset [MLBV10] is by far the most frequent (13), followed by the UMN dataset [MOS09] (6), Subway [ARSR08] and Avenue [LSJ13] datasets are equally frequent (5). In Table 4, we compare these four datasets in each row against ten attributes that we suggest. Only the UMN dataset is eligible for global abnormal event evaluation; in other datasets abnormal events are local. In general, datasets are small to medium-sized, shots are including indoor and outdoor environments with various camera distances, and most of the time camera shot is from a high angle. Typically the videos are in low quality due to the low resolution. Only the UCSD dataset provides pixel-perfect annotations to evaluate the anomaly localisation, which probably explains why many studies use it. Notably, the UCSD dataset has the highest crowd density, and Subway dataset has the least. Figure 7 illustrates samples of abnormal events from these datasets.

	Datasets					
Attributes	Subway [ARSR08]		UMN [MOS09]	UCSD [MLBV10]		Avenue [LSJ13]
	Entrance	Exit		Peds 1	Peds 2	
Abnormal events	Walking in wrong direction, loitering, no payment		Sudden crowd disperse	Strange pedestrian motion, non-pedestrian in the walkways		Person running, throwing objects, loitering
Data size	1 hour 36 minutes a long video	43 minutes long a video	4 minutes long video consists of 11 clips	70 image sequences: 34 for training, 36 for testing	28 image sequences: 16 for training, 12 for testing	37 videos: 16 for training, 21 for testing
Environment	Indoor		Indoor+Outdoor	Outdoor		Outdoor
Camera distance	Close to medium		Medium to long	Long		Medium
Camera angle	High		High	High		Eye-line
Video resolution	512 x 384		320 x 240	238 x 158	360 x 240	640x360
Crowdedness	Sparse		Relatively crowded	Crowded		Relatively crowded
Frame-level annotation	✔		✖	✔		✔
Pixel-level annotation	✖		✖	✔		Bounding-box
Accessibility	Upon request		✔	✔		✔

Table 4: Subway, UMN, UCSD, Avenue datasets are compared in each row against ten attributes, listed on the leftmost column [ARSR08, MOS09, MLBV10, LSJ13].

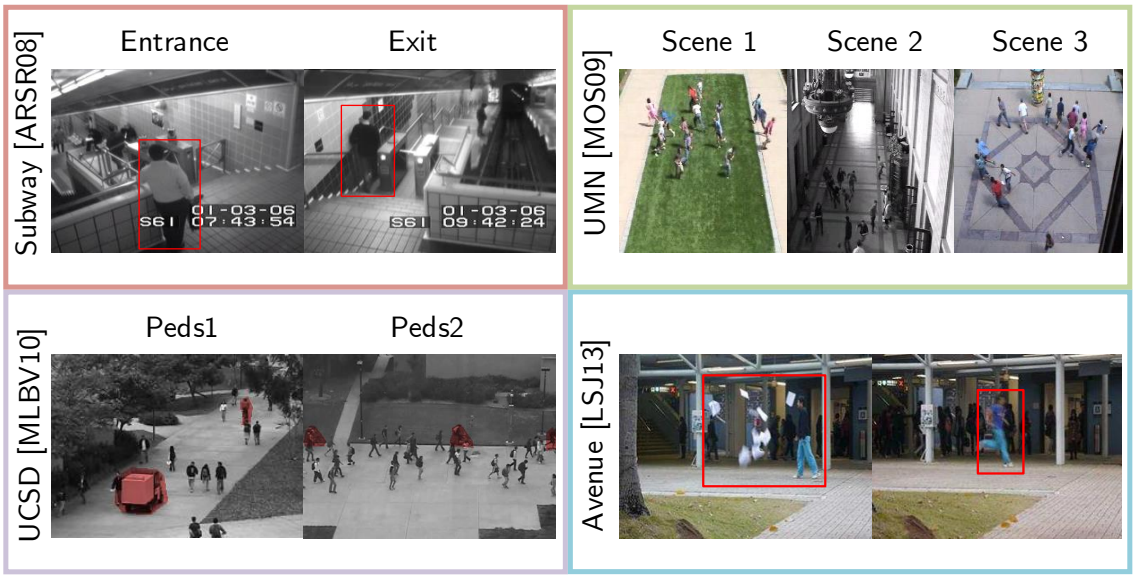

Figure 7: Example abnormal events from various datasets. Subway: loitering (Subway-entrance), wrong direction (Subway-exit) [ARSR08]. UMN: sudden crowd disperse in 3 different scenes [MOS09]. UCSD: non-pedestrians, car and bikers, in the walkways [MLBV10]. Avenue: a strange object is thrown and a person is suddenly running [LSJ13].

3.5 Review findings

The objective of this section was to get more insight into standard approaches in sparse representation based methods either utilised for feature learning or learning algorithm in abnormal event detection. In doing so, we reviewed 18 abnormal event detection studies, which we discussed in three sub-sections: event representation, classification frameworks, evaluation methodology.

Firstly, we note the effective representation of learned features, particularly deep learning-based features, and the state-of-the-art result demonstrated in combination with sparse representation based methods. Deep learned features generalise very effectively with more data and can be adapted to different contexts better than hand-crafted features. However, depending on applications, utilising deep learning techniques may not be feasible. The main arguments can be listed as follows: learning informative features require a large amount of training data, learning from data is computationally as well as financially expensive, and the last one is that difficulties concerning to interpretation deep learning models [GDL19]. We observe that hand-crafted features still have competitive abnormal detection performance compared to learned features. For example, Chu *et al.* [CXYC19] compares the abnormal detection performance of different features, and we see that the spatio-temporal gradient features are quite close to the deep learned features. Only the study by Xue *et al.* [XLJ16] use sparse coding in feature learning. On the other hand, nearly all studies use only normal activities from videos to learn events; only one study [HSS18] uses both abnormal and normal events.

Although the methods based on sparse representation work well in detecting abnormalities in a scene, most are yet to operate in real-time due to high testing cost, which is very crucial in video analysis. We note that only the algorithm by Lu *el al.* [LSJ13] shows promising results; the algorithm has high runtime performance with low hardware requirements and good classification result. However, we have observed high training time reported for this method in different studies [YLS17, GDL19].

On the other hand, certain studies do not include runtime performance analysis of their algorithms at all [XLJ16, YLZ$^+$16, SWXS18, LLXZ18, YFL18]. This certainly limits the comparative analysis in the research.

Many algorithms are evaluated with highly structured scenes, where the camera is usually stationary and shots taken from high angles, which is to imitate security

cameras. Additional research attention is needed to evaluate methods for the videos taken from different perspectives, such as videos taken from body cameras. Moreover, the demonstrated standard datasets do not have a variety or large enough data in order to evaluate the generalisability of methods. For example, only a small set of datasets are suitable for evaluating the localisation of abnormal events. It is not a surprise since manual pixel annotation is laborious work. Therefore, seeking for high-quality abnormal event detection datasets still remain as an open issue.

4 Methods

In this section, we describe our two-stage sparse representation based abnormal crowd event detection framework. Please note that we give the details of hyper-parameters settings in Section 5.1.

Firstly, in Section 4.1, we explain the feature extraction method that is used for event representation. We use the spatio-temporal gradient feature due to its effectiveness and simplicity reported in other studies [LSJ13, SFH16, YLS17, MH17, LSWJ18, ISPA18, GDL19].

Secondly, in Section 4.2, we describe our approach (the two-stage approach) in detail where we in the first stage, employ the k-means clustering algorithm to group features and remove the clusters with smaller samples. The idea is central to the abnormal event detection framework proposed by Ionescu *et al.* [ISPA18] and it has also motivated our work. However, we learn a dictionary for each cluster and use the sparse reconstruction error for abnormality detection. We hypothesise that utilising the k-means algorithm to remove smaller clusters and learning clustered features via dictionary learning could provide robust abnormality detection.

The overview of the method is illustrated in Figure 8:

- In the training stage, we extract spatio-temporal gradients and perform k-means to find different motion clusters. Then, we remove the clusters which have smaller samples. For each remaining cluster, we learn a dictionary using the approximate K-SVD algorithm [RZE08].
- In the testing stage, sparse codes against a new test instance are computed using the batch variant of the orthogonal matching pursuit algorithm [PRK93, RZE08] for all dictionaries that are learned in the training stage. The minimum reconstruction error is compared against a predefined threshold to predict abnormality of a feature.

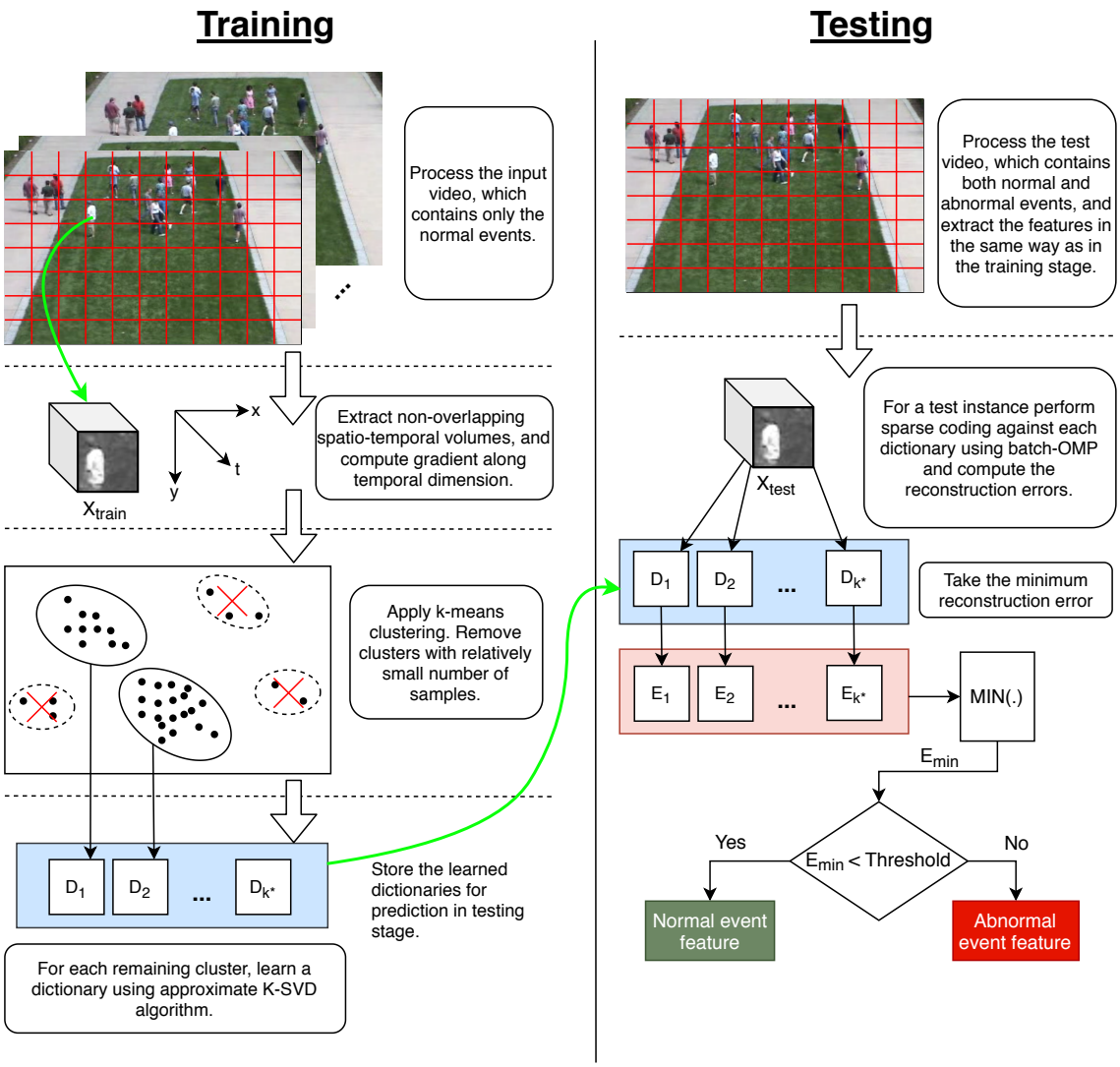

Figure 8: A visual depiction of our abnormal crowd event detection framework.

4.1 The spatio-temporal gradient feature

Motivated by many studies in the literature review [LSJ13, SFH16, YLS17, MH17, LSWJ18, ISPA18, GDL19], we use a spatio-temporal descriptor based on the 3D gradients in our framework. This type of feature was first utilised by Kratz and Nishino to model abnormal activities in an extremely crowded scene [KN09]. We will now present the definition of the feature according to them.

Given the input video, we first convert frames from colour images to grayscale in order to reduce the data dimension. Then, we uniformly partition each frame and extract a set of non-overlapping patches. All patches have the same spatial size of $l_x \times l_y$, in terms of pixels, where l_x is the width and l_y is the height of a patch. The resulting patches from l_t consecutive frames are stacked together to form a spatio-temporal volume of dimension $\mathbb{R}^{l_x \times l_y \times l_t}$. For a given volume V, the three dimensional spatio-temporal gradients are computed for each pixel i as:

$$\nabla V_i = [\frac{\partial V_i}{\partial x}, \frac{\partial V_i}{\partial y}, \frac{\partial V_i}{\partial t}], \tag{4.1}$$

where x, y, and t are the video's horizontal, vertical, and temporal dimensions, respectively [KN09]. The three-dimensional gradients represent both appearance ($\frac{\partial V}{\partial x}, \frac{\partial V}{\partial y}$) and motion ($\frac{\partial V}{\partial t}$) patterns of a local region [YLS17]. However, we use only the features computed along the temporal dimension of a video. We give an example visualisation of a spatio-temporal gradient feature applied to the entire image in Figure 9.

We follow the implementation provided by Del Giorno *et al.* [DGBH16], which is replicating the spatio-temporal gradient feature extraction scheme by Lu *et al.* [LSJ13]. The implementation[5] is based on the MATLAB computing environment; however, we implemented in the Python programming language. Unlike Lu *et al.* [LSJ13] we do not apply dimensionality reduction using the Principal Component Analysis (PCA) technique. Additionally, it is reported by Ionescu *et al.* [ISPA18] that for a spatio-temporal volume in $\mathbb{R}^{500}$ ($10 \times 10 \times 5$) using 100 first principal components of the gradient features do not affect the overall result. To be able to eliminate the static features (as they do not contain motion information), we apply a motion response threshold on the features. The sum of features of every sample is compared to a motion response threshold. However, this threshold value varies

[5]MATLAB implementation of spatio-temporal gradient feature extraction, accessed on 2019-11-24: `https://alliedel.github.io/anomalydetection/`

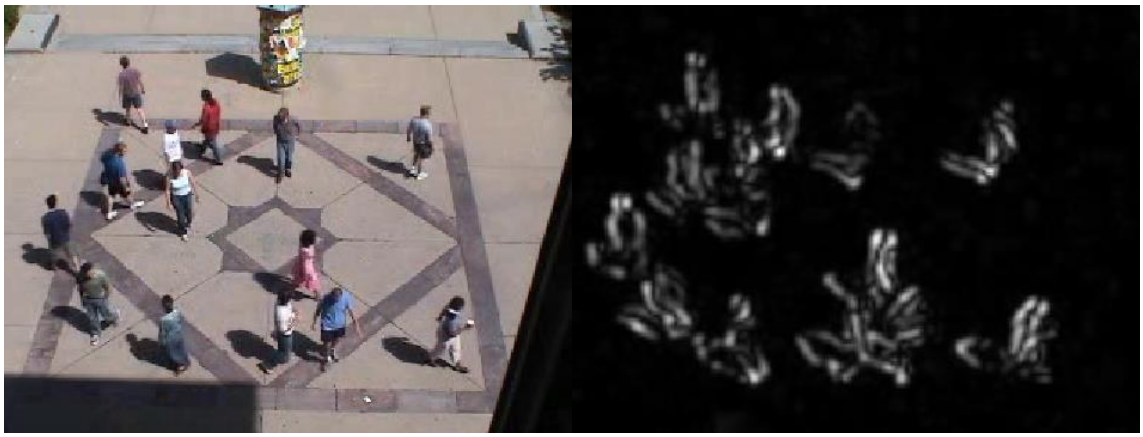

Figure 9: An example visualisation of a spatio-temporal gradient feature (right image) applied to the entire image from a scene in the UMN dataset (left image). On the right image, the white areas indicate motion or a change in pixel value.

depending on the size of the volume, and it is found empirically. For example, for a volume of size $10 \times 10 \times 5$ and its values in the range [0.0, 1.0], we used 5.0 sum pixel intensity as our motion response threshold.

Then, the resulting feature vectors are normalised using ℓ_2-norm to make all the features on a common scale. Here, we report a minor detail that this final pre-processing step in the given implementation by Del Giorno *et al.* [DGBH16] diverges from what is described in the original study by Lu *et al.* [LSJ13]. Instead of ℓ_2-norm normalisation, Lu *et al.* [LSJ13] applies standardisation (the feature vector has mean zero and standard deviation one).

The feature extraction process described here is applied identically at training and testing stages. In training, the input video is required to contain only normal events; and in testing, there is no such requirement. Another difference is that in the training algorithm, the features that are deemed to be noisy are eliminated by the k-means clustering algorithm before the dictionary learning algorithm. Whereas, in the testing algorithm, the features are not additionally clustered, but given directly as input to compute the sparse codes for prediction.

4.2 The two-stage sparse representation based method

Here we explain the two-stage sparse representation based classifier by first presenting the training algorithm and later the algorithm for testing a single feature.

Training algorithm

First stage. In the first stage of our training algorithm, the goal is to eliminate the noisy features and group similar motion features for the learning task. After we extract spatio-temporal gradient features, we use a clustering-based algorithm to achieve this.

There are various clustering methods available in the literature. Due to the high performance in data clustering, we employ the k-means clustering algorithm [Llo82] with k-means++ initialisation [AV07] as presented in Section 2. We have used the implementation from *scikit-learn*, the popular machine learning library in the Python programming language [PVG+11]. We experimentally decide the hyper-parameters for the number of clusters k, and the minimum number of data points in each cluster.

We assume noisy data points are far away from informative points in the feature space, and that when the number of clusters k is large, most clusters containing noisy points are sparse. So, once we cluster the data, we can eliminate the smaller clusters according to the designated threshold value (experimentally determined), and continue the learning task using the data in each remaining cluster. The same assumption lies in the basis of the method by Ionescu *et al.* [ISPA18]. We similarly illustrate this assumption on a synthetic dataset.

Using the two-dimensional Gaussian distribution, we generate 1000 data points from two Gaussians in different locations (standard deviation $\sigma = 0.8$). We partition the data into $k = 50$ clusters. Figure 10 shows the distribution of data points in each cluster. We select the threshold value relative to the sample size. Setting the cluster removal threshold to 14 data points eliminates 17 smaller clusters. In Figure 11, we illustrate clustered data points that are partitioned into 50 clusters. It is apparent in Figure 11 that the noisy features are well separated, and their corresponding clusters are sparse. Although we have illustrated here the assumption on a synthetic dataset, previous research validates the effectiveness of employing clustering prior to learning on a real-world application [ISPA18].

Second stage. Although the clustering algorithms alone can be used for abnormality detection, they are sensitive to high dimensional data. Therefore, after we remove the smaller clusters using the k-means clustering algorithm, the goal in this stage is to learn normal event patterns from each cluster. We formulate the learning

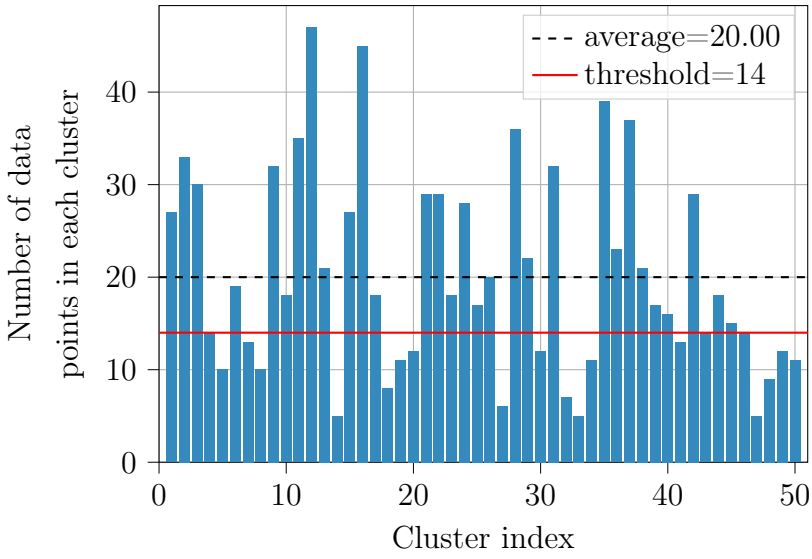

Figure 10: Distribution of data points in each cluster.

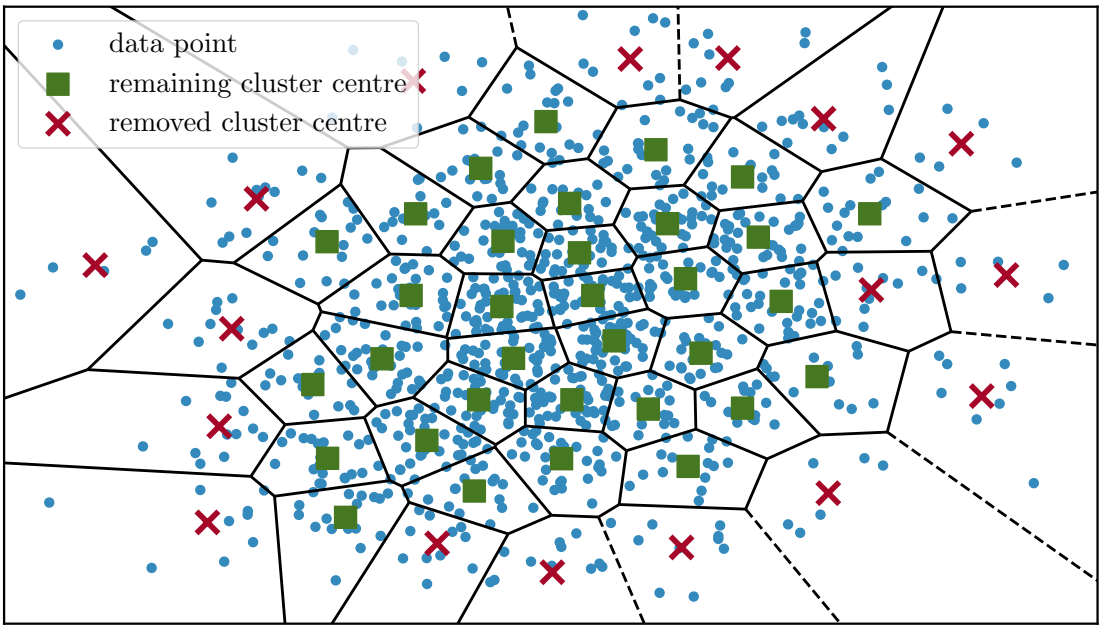

Figure 11: Clustering result of 1000 data points into 50 clusters. The data points in each cluster are depicted in Voronoi regions. Removed and remaining clusters can be distinguished from the figure according to their cluster centres.

task as a sparse dictionary learning problem that is presented in Section 2.

Let S be the clusters that are determined initially, and let S^* denote the remaining

clusters in the first stage after removing according to a threshold, where k^* is the number of remaining clusters. Also, let $X = [x_1, ..., x_N] \in \mathbb{R}^{P \times N}$ denote spatio-temporal gradient features in cluster $s_j \in S^*$. We learn a normal event dictionary $D = [d_1, ..., d_K]$ for each cluster $s_j \in S^*$ with a target sparsity ζ, as defined in (2.7), where K is the number of atoms in the dictionary. Dictionary atoms are randomly initialised from the input data. After dictionary initialisation, the approximate K-SVD algorithm [RZE08] is used to optimise the dictionary and sparse coefficients. Finally, the learned normal event dictionaries for each cluster are stored for testing. We give the pseudo-code for the training algorithm in Algorithm 6.

We first implemented the approximate K-SVD algorithm in Python and used the Python implementation of the batch-OMP algorithm from the scikit-learn library. However, the Python implementation ran quite slowly (7-15 FPS). Therefore, we used a highly-optimised, parallel, C++ implementation of the approximate K-SVD algorithm, which is available open-source. This has greatly improved the run time performance of our framework (60-300 FPS). The implementation also provides Python bindings, so that we used it in our framework without much change. However, at the time of writing, there were some dependency and compatibility issues with the implementation. We contributed to it by fixing the issues, and used this implementation throughout in the experiments[6].

Algorithm 6 Training algorithm

Input: Clusters $S^* = \{s_1, ..., s_{k^*}\}$, target sparsity ζ
Output: Learned dictionary set $D_{\text{out}} = \{D_1, ..., D_{k^*}\}$
$D_{\text{out}} = \varnothing$
for each $s_j \in S^*$ **do**
    Initialise dictionary $D_j \in \mathbb{R}^{P \times K}$ using data points in s_j.
    Optimise the dictionary and the sparse coefficients, as defined in (2.7), using the approximate K-SVD [RZE08]. ▷ See Algorithm 4.
    $D_{\text{out}} = D_{\text{out}} \cup \{D_j\}$
end for
return D_{out}

[6]Fork of the efficient implementation of the approximate K-SVD algorithm, accessed on 2019-11-25: https://github.com/eozer/pyksvd

Testing algorithm for a single feature

In the final step of our abnormal detection framework the goal is to compute the sparse coefficient vector a for a new test instance $x_{\text{test}} \in \mathbb{R}^P$ using the learned dictionaries from set D_{out}. The general assumption in sparsity-based abnormal detection is that the sparse coefficient vector is dense for abnormal instance. Whereas, it is sparse for a normal instance. Therefore, decision for x_{test} can be made according to the reconstruction error from $\|x_{\text{test}} - Da\|_2^2$, and a given threshold T.

Pseudo-code of the testing algorithm for a single spatio-temporal gradient feature is given in Algorithm 7. For each learned dictionary in D_{out}, we compute the sparse coefficients of x, and its reconstruction error (line 6). In computing the sparse coefficients, we use a greedy method, a variant of orthogonal matching pursuit algorithm, called batch-OMP algorithm by Rubinstein *et al.* [RZE08]. The minimum reconstruction error is used to decide whether a test instance is abnormal or normal. We scale the reconstruction errors into the range [0.0, 1.0]. A higher value means that a feature has a higher abnormality score. Lastly, we follow a set of post-processing techniques, similar to previous studies [LSJ13, DGBH16, ISPA18]. We smooth the obtained predictions using a three-dimensional uniform filter. We use the multi-dimensional uniform filter implementation (`scipy.ndimage.uniform_filter`) from *SciPy* package [JOP01].

Algorithm 7 Testing a single feature with the two-stage method.

1: **Input:** Input feature $x \in \mathbb{R}^P$, $D_{\text{out}} = \{D_1, ..., D_{k^*}\}$, threshold T, target sparsity ζ
2: **Output:** Reconstruction error E_{min}
3: $E = \varnothing$ ▷ Initialise reconstruction errors
4: **for each** $D_j \in D_{\text{out}}$ **do**
   Find sparse codes A using OMP: ▷ See Algorithm 1.
5:
$$a_j = \arg\min_a \left\|x - D_j a\right\|_2^2 \text{ subject to } \quad \|a\|_0 \leq \zeta \tag{2.4}$$
6: $E_j = \left\|x - D_j a_j\right\|_2^2$ ▷ Compute the reconstruction error
7: $E = E \cup \{E_j\}$
8: **end for**
9: $E_{\text{min}} = \min(E)$ ▷ Take only the minimum reconstruction error
10: **return** E_{min}

In order to get frame-level abnormal decisions, we apply a set of post-processing steps. First, we aggregate the scaled reconstruction errors of every feature in a frame and take the highest one. For example, for an image size of 160×120 (width×height) extracting 10×10 non-overlapping patches yields 192 sub-regions. So for 192 abnormal predictions, we take the highest value. Next, we apply the one-dimensional Gaussian filter along the temporal dimension to smooth the frame-level abnormality scores. We use the one-dimensional Gaussian filter implementation (`scipy.ndimage.gaussian_filter1d`) similarly from *SciPy* package [JOP01]. Then, we scale the frame-level abnormal predictions into the range [0.0, 1.0]. In this case, a higher value indicates that a frame has a higher abnormality score. Lastly, a frame-level threshold is used to decide whether a frame is abnormal or not.

5 Experiments and results

In this section, we conduct extensive experiments and report the results of our abnormal detection framework. In Section 5.1, we describe the used dataset, common hyper-parameters, evaluation criteria, and experiment workstation. In Section 5.2, we present the initial experiment result based on common hyper-parameters used from the literature. In Section 5.3, we analyse the effects of different hyper-parameters on the performance of the method. Finally, we compare our approach against different methods in Section 5.4.

5.1 Experiment settings

UMN dataset

We are mainly interested in understanding global abnormal events, such as an escape scenario or a panic situation in a crowd. Therefore, we used the UMN dataset in all our experiments [MOS09]. The UMN dataset consists of three different scenes, a total of 11 video clips. There are 7740 frames (scene 1: 1453, scene 2: 4144, scene 3: 2143), which is an approximately 4 minutes long video. The dataset has a resolution of 320×240 (width×height) pixels. The normal events are people walking around randomly, and the abnormal events are sudden crowd dispersal, imitating a panic situation. We illustrate a set of samples of each scene from the UMN dataset in Figure 12.

To train and test our algorithm, we followed the common practice, where first N frames from each scene were used for training, and the remaining frames were used for testing [CYL13, HFWW13, YLZ+16, LSWJ18, ISPA18]. However, there is no consensus on how many frames to choose [CYL13, HFWW13, YLZ+16, LSWJ18]. For example, Cong *et al.* [CYL13] use first 400 frames from each scene. Whereas, Han *et al.* use 30, 60, 90 from scene 1—3, respectively [HFWW13]. Therefore, we followed the approach by Yi *et al.* [YLZ+16], and used 400, 300, 400 frames from scene 1—3, respectively. By using these numbers of frames, we use specifically normal events while retaining a high number of features from each video clip in the training stage.

The UMN dataset provides frame-level annotations embedded in the video. Due to the lack of text-based frame-level annotations, we document the ground truth annotations of abnormal frames for each video clip in Table 5. Note that to use

Figure 12: Sample normal and abnormal events from the UMN dataset.

the annotations from Table 5 one needs to first convert the UMN dataset from a video format to an image format, and rename the frames starting from one to the end of a video clip. The processed dataset can be downloaded from the link `http://tiny.cc/processed_umndset` (accessed on 2020-02-06).

	Video clip	**Frame numbers**	
		Start	**End**
Scene 1	01	526	615
	02	706	815
Scene 2	03	354	533
	04	604	683
	05	533	742
	06	484	563
	07	774	895
	08	494	667
Scene 3	09	600	639
	10	630	659
	11	770	789

Table 5: The frame-level annotations of abnormal frames of the UMN dataset.

Common hyper-parameters

In our initial experiment, we used common hyper-parameters settings from the literature [LSJ13, XLJ16, LSWJ18, ISPA18]. Unless otherwise stated, the hyper-parameters mentioned here can be assumed to be fixed within all our experiments.

We resized each frame from the UMN dataset to 160×120 pixels and extracted uniform non-overlapping patches of size 10×10 pixels. The patches from 5 consecutive frames were stacked together to form spatio-temporal volumes, and then we took gradient along the temporal dimension of spatio-temporal volumes. Motion response threshold was set to 5.0 sum pixel intensity to remove all static features. We applied ℓ_2-norm normalisation to each feature individually. The final feature vector had a dimension of 500. We processed the UMN dataset video with a frame stride (*i.e.*, how many frames are skipped) of 1 in order to generate as many features as possible.

Ionescu *et al.* choose the number of clusters k always proportional to the training data [ISPA18]. They restrict this so that there are on average 1000 features per cluster, and they remove the clusters with less than 500 features. We used a predefined number of clusters, and the cluster removal was controlled by a percentage value (instead of a fixed number of samples) to be able to adapt the changes of sample sizes dynamically. For example, setting the cluster removal percentage to 25% means that we remove at least 25% of the smaller clusters, depending on the distribution of samples in all clusters. We experimentally selected 20 clusters, and removed 40% of clusters. We restarted the k-means algorithm 10 times and took the best cluster assignments. For a single run of the k-means algorithm, maximum iteration was set to 300. In further sections, we also report the effect of different cluster sizes and cluster removal percentages.

For each remaining cluster, we learned a dictionary with 2000 atoms. Sparsity term ζ was set to 1. Maximum iteration for optimising each dictionary was set to 500 iterations. Convergence tolerance of a dictionary was set to 10^{-8}. If a dictionary convergence was reached, we stopped early and did not continue for further iterations.

Lastly, the post-processing parameters were as following; the three-dimensional uniform filter size was set to 3 and the standard deviation for one-dimensional Gaussian filter was set to 5.0.

Evaluation criteria

In this thesis, we used only frame-level evaluation criteria. An algorithm was used to determine if frames contain normal or abnormal events. The result was then compared to the frame-level ground-truth annotations for each frame. The true positive and false positive rates were calculated accordingly. We computed the true positive rate versus the false positive rate for different threshold values in order to draw a receiver operating characteristic curve. Finally, the performance was summarised using the area under the receiver operating characteristic curve. In all experiments, we have optimised for the area under the receiver operating characteristic curve.

In addition, we used the classification accuracy and the F_1 score in order to provide a simple and intuitively easy way to interpret the performance of methods. We used Youden's J statistics to calculate optimum frame-level threshold index from receiver operating characteristic curve, given by

$$t_{optidx} = \arg\max \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} - \frac{\text{false positives}}{\text{false positives} + \text{true negatives}}. \quad (5.1)$$

Furthermore, the classification accuracy and the F_1 score were calculated accordingly. See Table 2 for the definitions of the aforementioned metrics. The runtime performance is reported in terms of frames per second. We report end-to-end prediction pipeline speed, which includes feature extraction, prediction, and post-processing steps. We also report the evaluation results for each scene separately. Results for the UMN dataset were calculated by averaging the results of all the scenes, and expressed as 'Overall' in figures and tables. We used the macro-average method to simplify the calculation of overall results.

Experiment workstation

All the experiments were performed using Python version 3.6.10 on an Intel Xeon 2.40 GHz processor with 32 GBs memory. We used the Intel Math Kernel library (Intel MKL) to leverage the optimised math routines. The performance impact without Intel MKL may be a subject to change. But, we observed between 5—10 times performance gain by compiling Intel MKL and linking to several Python libraries. The five most frequently used Python libraries in our experiments were *OpenCV* (version 4.1.2.30) [Bra00], SciPy (version 1.3.1) [JOP01], *NumPy* (version 1.18.1) [WCV11], scikit-learn (version 0.22.1) [PVG+11], and *PyOD* (version 0.7.7) [ZNL19].

5.2 Initial experiment

Using the hyper-parameters introduced in Section 5.1, our abnormal event detection framework took approximately two hours to train and test for the UMN dataset. In the following pages, we first present the qualitative result of our framework and demonstrate for each scene frame-level detection results as well as the spatial location of abnormal events on testing videos. We then provide the quantitative results on the performance of our framework.

Qualitative results on UMN scene 1

We start by reporting the number of samples in the training and testing stages. In training, we obtained a total of 76800 samples after the feature extraction process. Setting the motion response threshold to 5.0 sum pixel intensity, removed a large number of samples, 71616. Only the remaining 5184 samples were used for the training stage. The training took approximately 10 minutes to finish. Moving on to the testing stage, we got 200448 samples after the feature extraction, where 183684 samples were removed similarly by the motion response threshold. This means our method made predictions only for the remaining 16764 samples, and the removed samples were determined already as normal. The runtime performance was on average 101 FPS.

In Figure 13, we illustrate the frame-level abnormality scores on Video02 in the UMN dataset. Abnormal frame threshold was calculated as 0.39 for Video02 by Equation 5.1. The smoothed abnormality scores, as shown in Figure 13b, were the final results of our method for decision making on Video02. We select a set of frames (260, 670, and 715) from Figure 13c to take a closer look at the spatial location of some normal and abnormal events. The corresponding results by our approach are visualised in Figure 14. Due to the nature of our problem being imbalanced, more frames were classified as normal than abnormal. However, our method can generally understand abnormal crowd events in scene 1. For example, we illustrate a true positive abnormal crowd activity detection result in Figures 14e and 14f. We observe a few findings according to these results.

First, the post-processing steps have a positive effect on our abnormal event detection framework. We illustrate the frame-level without and with the post-processing steps in Figures 13a and 13b, respectively. The most obvious difference between these two figures is fluctuation in the scores. According to the figure, the post-

processing steps improved stability and detection performance as it smoothed out sudden changes. For example, frame 260 is above the threshold in Figure 13a, and in 13b it is just below the threshold. We visualise the abnormality scores of the corresponding features of frame 260 in Figure 14a, and next to it 14b shows the final detection result. Due to smoothing, frame 260 was classified as a true negative (normal event), and so that the abnormal event alert was not triggered. In fact, one can observe the same pattern on different frames.

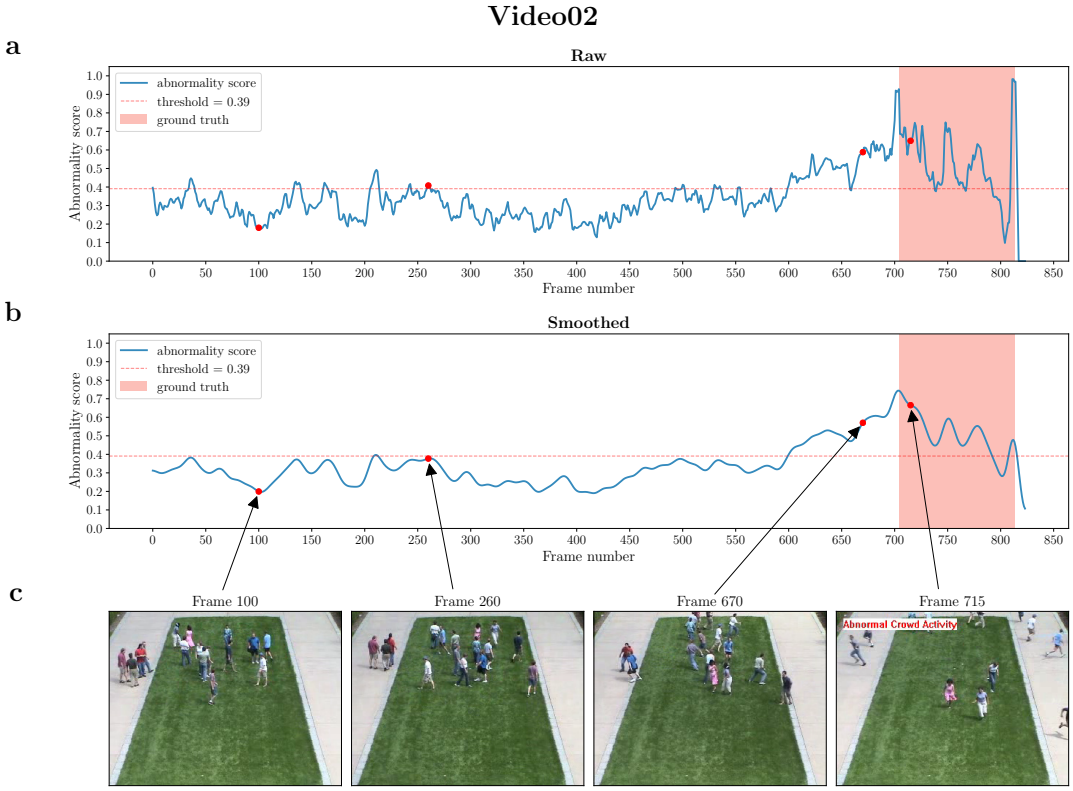

Figure 13: The frame-level abnormality scores provided by our framework on Video02, UMN scene 1. **(a)** Abnormality scores as a function of a frame number, without the post-processing steps are applied, denoted by solid lines while shaded red regions denote ground truth. **(b)** Same as (a), but including the post-processing steps. The abnormal threshold is computed as 0.39 for Video02, which is denoted by the dashed red line. **(c)** Frames 100, 335, 670 and 715 are shown as some examples from Video02. The corresponding abnormality scores of the frames are annotated as red points in (a) and (b).

Second, the frame-level ground truth in the UMN dataset seems to be defined with a certain assumption in mind. From this standpoint, our approach failed in some cases. For example, our method generated numerous false positives, specifically

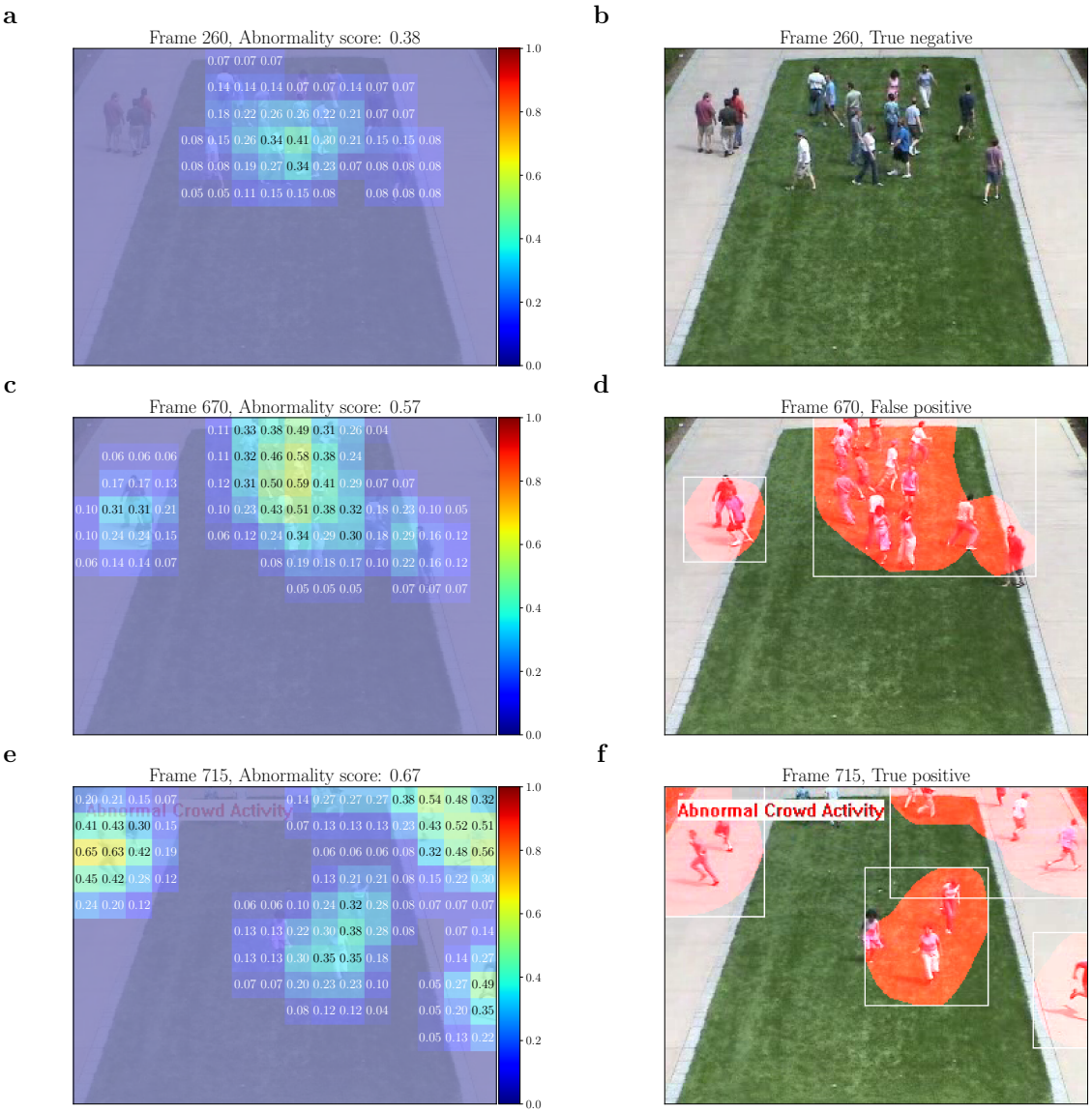

Figure 14: Spatial location of some normal and abnormal events determined by our framework on Video02, UMN scene 1. Abnormal event threshold is set at 0.39 for Video02. Red regions with white bounding boxes indicate an alert triggered by our framework. **(a), (b)** present a true negative on Frame 335. **(c), (d)** present a false positive on Frame 670 due to early warning. **(e), (f)** present a true positive on Frame 715. Annotated texts, as well as the colour maps in (a), (b), (c) indicate an abnormality score of a spatio-temporal gradient feature. The severity of an abnormal feature is illustrated by a higher value and being closer to red colour on the colour map.

between frame 670 and 704, where the abnormality score kept rising starting from

0.57. We show an event localisation result of frame 670 (a false positive prediction) in Figures 14c and 14d. When we further examine this, it is difficult to validate whether the frames belong to normal frame class, because the escape situation seems to be starting around frame 670. Nevertheless, we conclude that our method failed in these cases due to the early warning.

Qualitative results on UMN scene 2

We similarly begin by giving an overview of training and test stages in scene 2. In training, we got 230400 samples after the feature extraction, 226902 of which were removed and the remaining 3498 samples were used for training. Despite the number of samples after the feature extraction, the number of samples that were used in the training stage of UMN scene 2 was even less than scene 1. This implies there is less crowd movements in scene 2. Training took approximately 8 minutes. In testing, we obtained a total of 559680 samples, 537762 were removed, and prediction was done only for the remaining 21918 samples. The runtime performance on scene 2 was on average 226 FPS.

In Figure 15, we illustrate the frame-level abnormality scores on Video07 in the UMN dataset. Here, our method performed poorly compared to scene 1. Several spikes can be observed from the raw output in Figure 15a, where no post-processing was applied. Even smoothing does not seem to help much. It can be seen from Figure 15b that the abnormality score decreases towards the end of the video as the scene becomes empty. The abnormal frame-level threshold was computed as low as 0.27 (See Eq. 5.1), and as a result, our method generated large amounts of false positives.

We select a set of frames (210, 750, and 825) from Figure 15c, and look closer into the detection results in Figure 16. We illustrate two false positive cases in Figures 16a, 16b and 16c, 16d. Specifically, Figures 16a, 16b illustrating a false positive detection result belonging to frame 210. Observe the high abnormality score of a patch in Figure 16a, which caused a false alert. The reason seems to be because of sudden illumination changes that occur as a result of a person entering the scene. Methods by previous research also generate the same false alert on this event [ISPA18]. Next, in Figures 16c and 16d, we see that our method alerted early again according to the ground truth, and therefore we classified frame 750 as false positive. Nevertheless, our method was still be able to detect the abnormal crowd activity on Video07. We show an example detection result from frame 825 in Figures

16e and 16f.

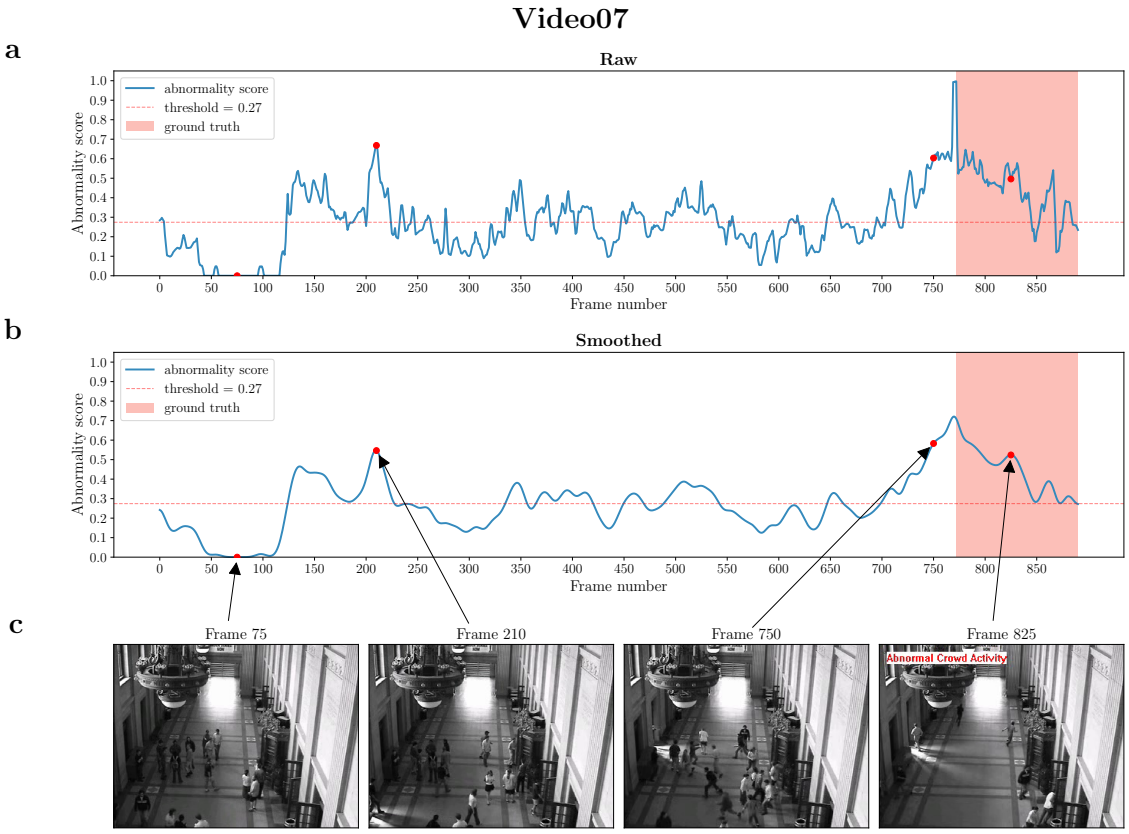

Figure 15: The frame-level abnormality scores provided by our framework on Video07, UMN scene 2. **(a)** Abnormality scores as a function of a frame number, without the post-processing steps are applied, denoted by the solid lines; while shaded red regions denote ground truth. **(b)** Same as (a), but including the post-processing steps. The abnormal threshold is computed as 0.27 for Video07, which is denoted by the dashed red line. **(c)** Frames 75, 210, 750 and 825 are shown as some examples from Video07. The corresponding abnormality scores of the frames are annotated as red points in (a) and (b).

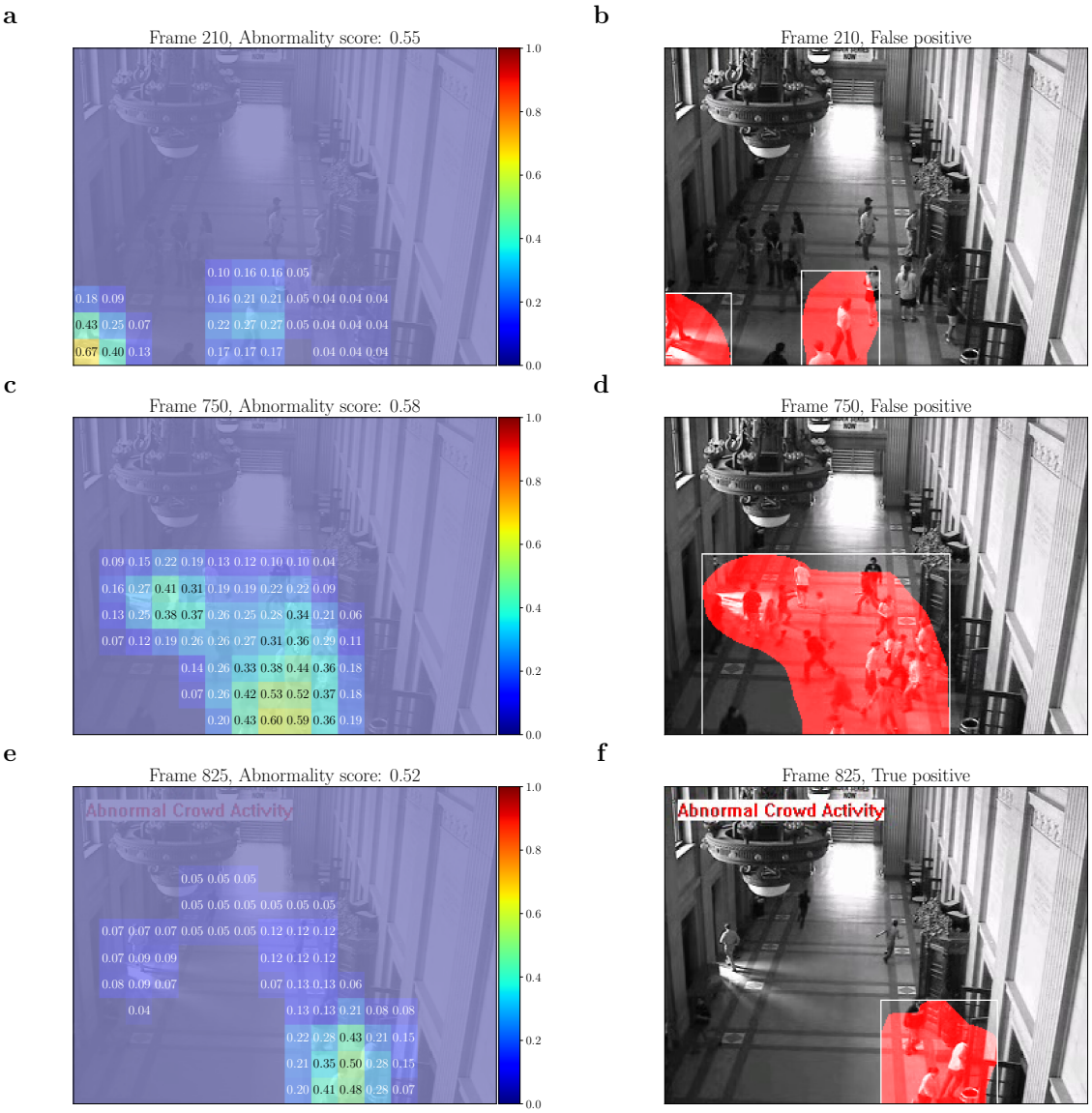

Figure 16: Spatial location of some abnormal events determined by our framework on Video07, UMN scene 2. Abnormal event threshold is set at 0.27 for Video07. Red regions with white bounding boxes indicate an alert triggered by our framework. **(a), (b)** present a false positive on Frame 210 due to a person entering to hall and lightning change. **(c), (d)** present a false positive on Frame 750 due to early warning. **(e), (f)** present a true positive on Frame 825. Annotated texts, as well as the colour maps in (a), (b), (c) indicate an abnormality score of a spatio-temporal gradient feature. The severity of an abnormal feature is illustrated by a higher value and being closer to red colour on the colour map.

Qualitative results on UMN scene 3

Overview of the training and testing stages for UMN scene 3 was as follows. In training, we got 76800 samples after the feature extraction, 65316 of which were removed and the remaining 11484 samples were used for training. Training took approximately 1 hour 36 minutes, which was more than the combination of scene 1 and scene 2. In testing, we obtained a total of 332160 samples, 276286 were removed, and prediction was done only for the remaining 55874 samples. The runtime performance on scene 3 was on average 60 FPS, being the lowest one compared to the first two scenes.

In Figure 17, we illustrate the frame-level abnormality scores on Video11 in the UMN dataset. First, we observe the abnormality score on average is the highest among all scenes regardless of the effect of the post-processing steps, as shown in Figures 17a and 17b. We note that the ground truth region of Video11 is much tighter than earlier test videos. However, there is a clear difference in the case of abnormal crowd activity. With the computed threshold value 0.60, our method can clearly separate the normal and abnormal events.

We select a set of frames (275, 730 and 770) from Figure 17c. We present a true negative, a false positive, and a true positive classification result in Figure 18. Despite the density (in small region there are momentarily 7 people) and the constant movement of the crowd in Figure 18a and 18b, our framework classified a normal event correctly. The density and fast-paced crowd movement may also explain why, on average, the frame-level abnormality score was higher on scene 3 than the previous two scenes. Next, we illustrate a false positive example that belongs to frame 730 in Figure 18c and 18d. Similarly in previous scenes, our method made an early detection according to the ground truth, and therefore the result is false positive. However, this time it is obvious from frame 730 that there is a sudden crowd dispersal. Finally, we illustrate a true positive detection in Figures 18e and 18f. Starting from frame 770 it can be seen that our method correctly detected abnormal crowd activity.

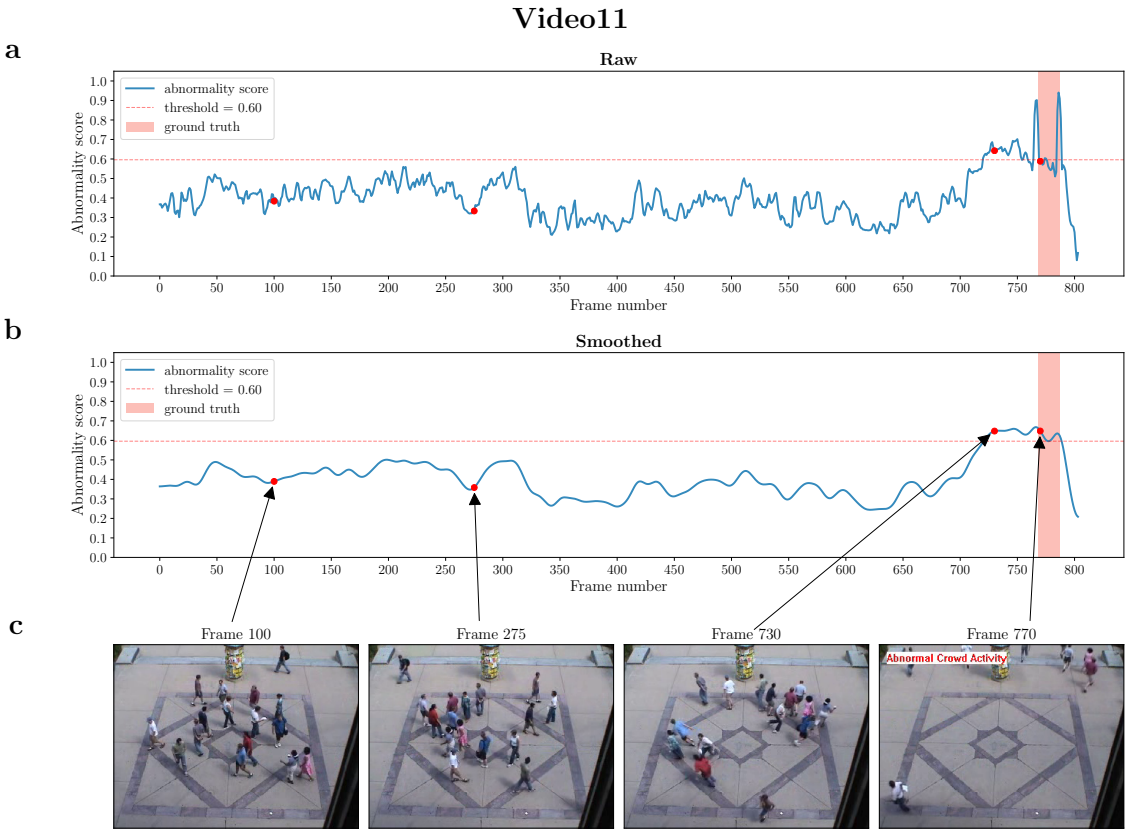

Figure 17: The frame-level abnormality scores provided by our framework on Video11, UMN scene 3. **(a)** Abnormality scores as a function of a frame number, without the post-processing steps are applied, denoted by solid lines while shaded red regions denote ground truth. **(b)** Same as (a), but including the post-processing steps. The abnormal threshold is computed as 0.60 for Video11, which is denoted by the dashed red line. **(c)** Frames 100, 275, 730 and 770 are shown as some examples from Video11. The corresponding abnormality scores of the frames are annotated as red points in (a) and (b).

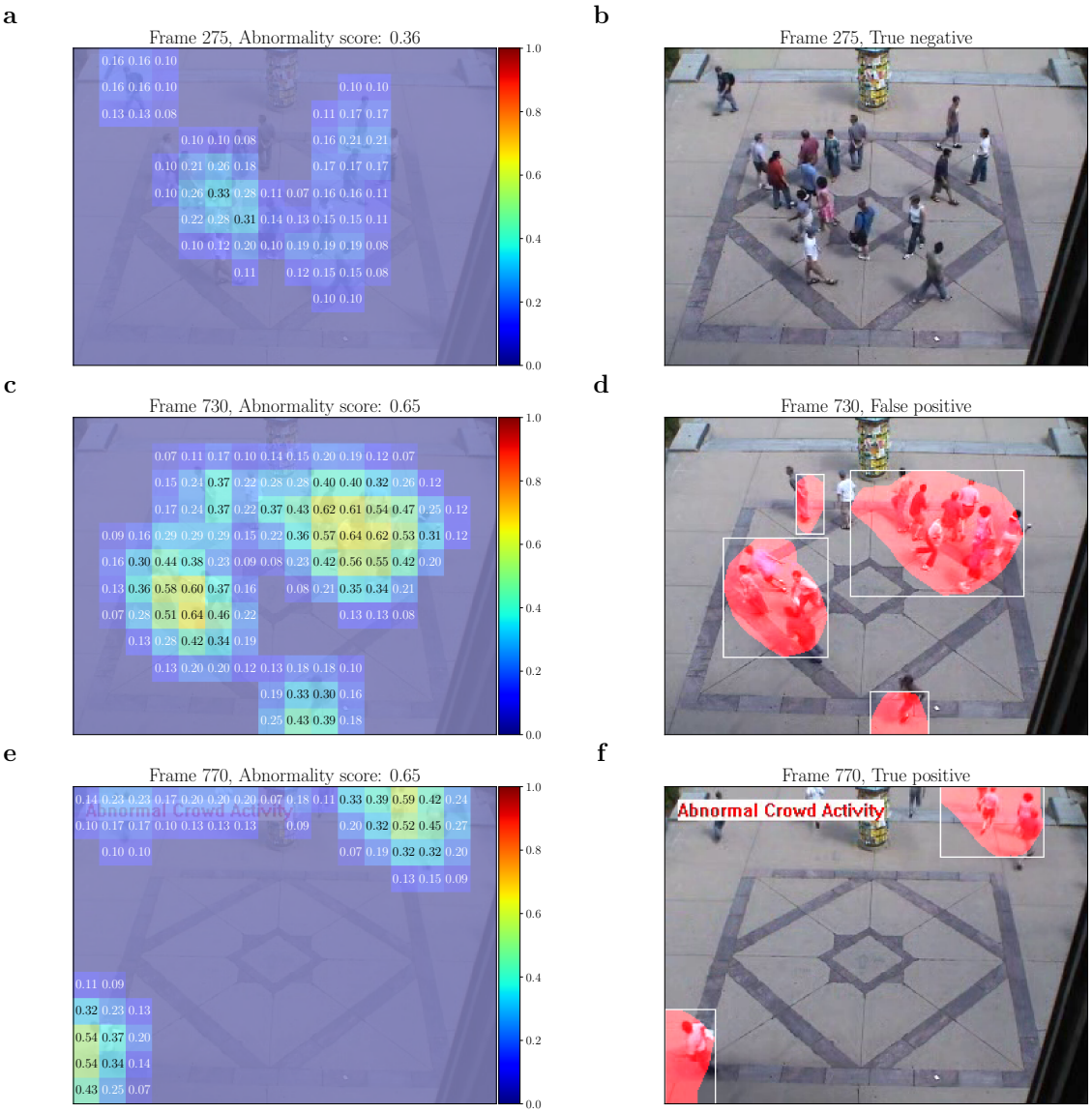

Figure 18: Spatial location of some normal and abnormal events determined by our framework on Video11, UMN scene 3. Abnormal event threshold is set at 0.60 for Video11. Red regions with white bounding boxes indicate an alert triggered by our framework. **(a), (b)** present a true negative on Frame 275. **(c), (d)** present a false positive on Frame 730 due to early warning. **(e), (f)** present a true positive on Frame 770. Annotated texts, as well as the colour maps in (a), (b), (c), indicate an abnormality score of a spatio-temporal gradient feature. The severity of an abnormal feature is illustrated by a higher value and being closer to red colour on the colour map.

Quantitative results

Here, we quantitatively evaluate the performance of our framework in terms of ROC curve, AUC score, classification accuracy, and F_1 score.

We present ROC curves and AUC scores of our method on each UMN scene and overall UMN dataset in Figure 19. Scene-wise abnormal frame thresholds were calculated according to the ROC curves, and they were equal to 0.40, 0.36, and 0.51 for UMN scene 1-3, respectively. We note that these thresholds are different from what we have shown previously. The closer a ROC curve of a model is to the upper-left corner of a figure, the better it performs. According to this, our method performed best on UMN scene 3 with 0.89 AUC score. It has a reasonably high detection rate when the false positive rate is low. Furthermore, the worst result obtained on UMN scene 2 with 0.66 AUC. Observe how the ROC curve of our model on UMN scene 2 is close to a random guess. There is only 0.16 AUC score difference. Overall, our method achieved 0.78 AUC on the UMN dataset.

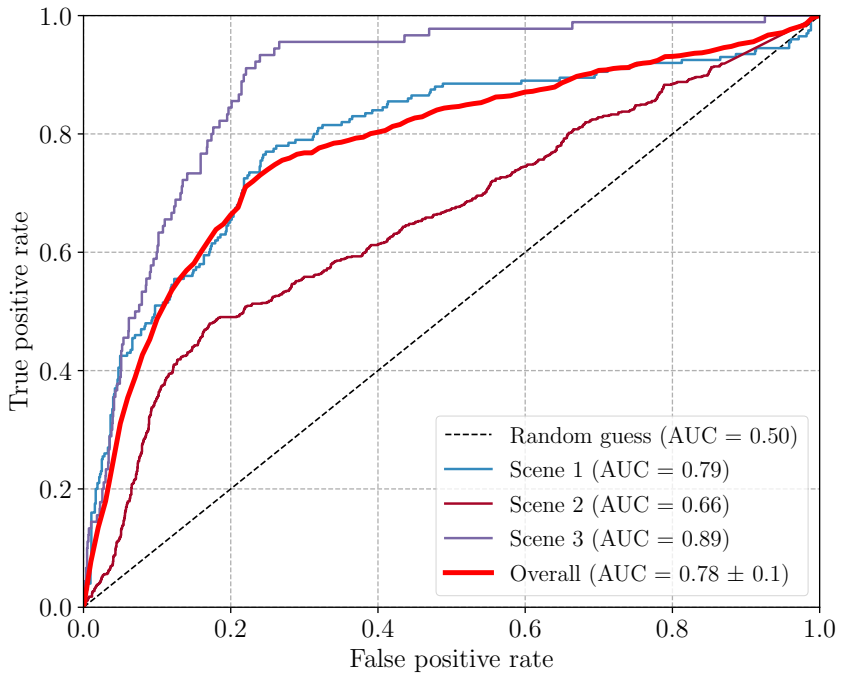

Figure 19: ROC curves and AUC scores (given in parenthesis) of our method on UMN scene 1 (blue), scene 2 (brown), scene 3 (purple), and overall UMN dataset (red). ROC curve of a random guess is shown as dashed black lines.

In addition, we present normalised confusion matrix and performance metrics of our method on each UMN scene and overall UMN dataset in Figure 20. Despite the differences in AUC scores, observe how the classification accuracies are similar in

each UMN scene. Our method achieved 0.751 classification accuracy on the UMN dataset; scene 3 was the easiest one. However, observe also the F_1 score on UMN scene 3, which is equal to 0.293 being the worst among all scenes. Low F_1 score can be explained by the fact that there are only small numbers of abnormal class instances in UMN scene 3. As a result, there are less true positive data being classified.

In summary, we get an unsatisfactory result in our initial experiments compared to previous research. For example, a pure optical flow approach achieves 0.84 AUC on the UMN dataset [MOS09]. However, this also indicates that there is room for improvement. Better results may be obtainable by setting different hyper-parameters.

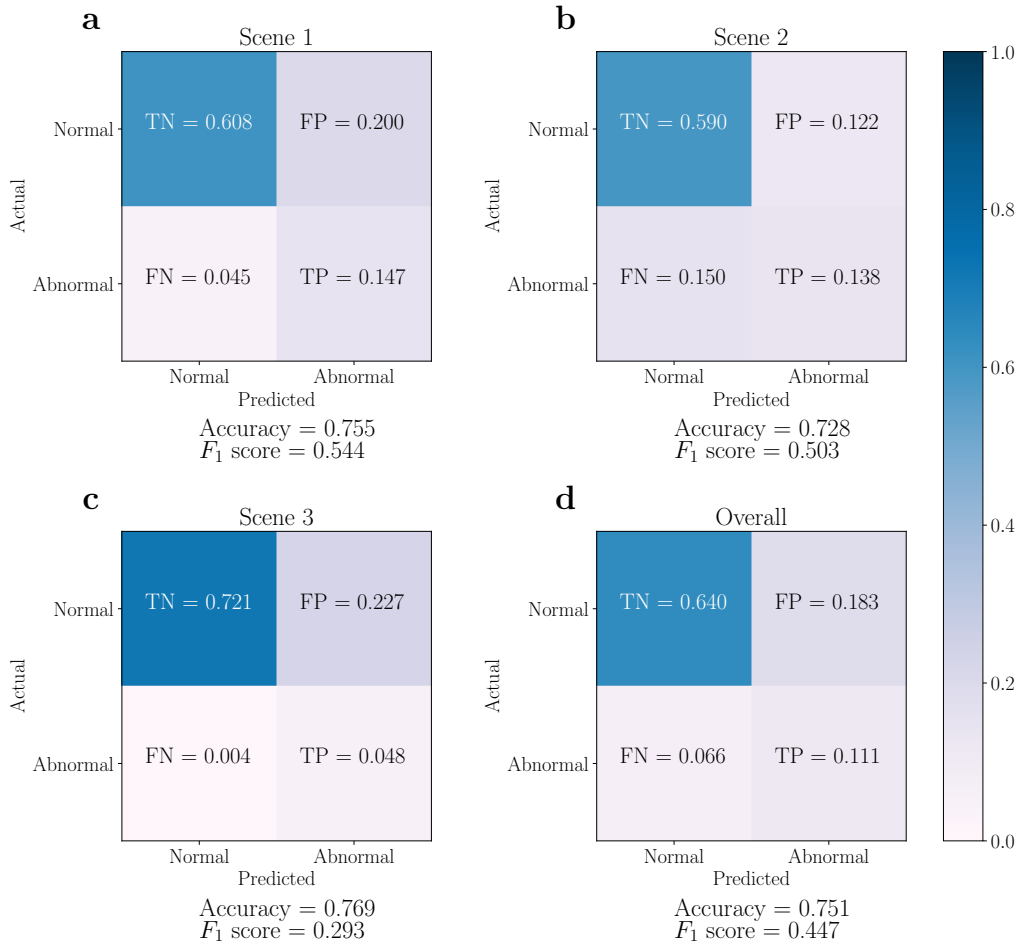

Figure 20: Normalised confusion matrix and performance metrics of our framework on UMN scene 1 **(a)**, scene 2 **(b)**, scene 3 **(c)**, and overall UMN dataset **(d)**. Colour map of confusion matrices for abnormal event classification is presented on the right. Darker blue colour means more data is being classified in that class.

5.3 Effect of hyper-parameters

We explore the following hyper-parameters that may improve abnormal event detection result: number of dictionary atoms (K); sparsity term (ζ); cluster size and cluster removal percentage; motion response threshold; post-processing filters; image size, patch size, and motion response threshold. We used the grid search method to perform hyper-parameter optimisation. A parameter under consideration was exhaustively searched from a pre-defined search space, while the rest of the hyper-parameters were fixed according to the common hyper-parameters introduced in Section 5.1.

Number of dictionary atoms (K). Complex crowd activities may demand more dictionary atoms for a better representation. With the increased number of dictionary atoms (K), we may improve the AUC. We illustrate the change of AUC with respect to the number of dictionary atoms (K) on the UMN dataset in Figure 21. The search domain for K was set to $500 \leq K \leq 5000$ with a linear increment step of 500 atoms. It can be seen from the figure that a larger dictionary does not always improve AUC. Luo *et al.* have a similar observation about the effect of dictionary size in their method on different datasets [LLG17]. Moreover, the difference in AUC for each scene is negligibly low. There is a slight AUC improvement with the peak at 1500 dictionary atoms on UMN scene 1 and 2. However, as K was increased in each cluster, the training time was also significantly increased due to more iterations in the dictionary update step. In conclusion, the dictionary atom size has a minimal effect on the performance of our framework.

Sparsity term (ζ). ζ controls the sparsity of solution in Equation (2.4); the smaller ζ, the more sparse it is. ζ was changed according to the search space $1 \leq \zeta \leq 10$. Observe how this hyper-parameter affects the method in Figure 22. No clear correlation was found between AUC and ζ. There is a general increase in AUC on UMN scene 2 and 3. On the other hand, AUC decreases in UMN scene 1. It can be seen that the optimal number of ζ varies in each scene, and on the UMN dataset it equals to 7 non-zero coefficients. However, the maximum AUC difference is approximately 0.005. It indicates that the effect of sparsity term (ζ) in AUC is again too small on the UMN dataset.

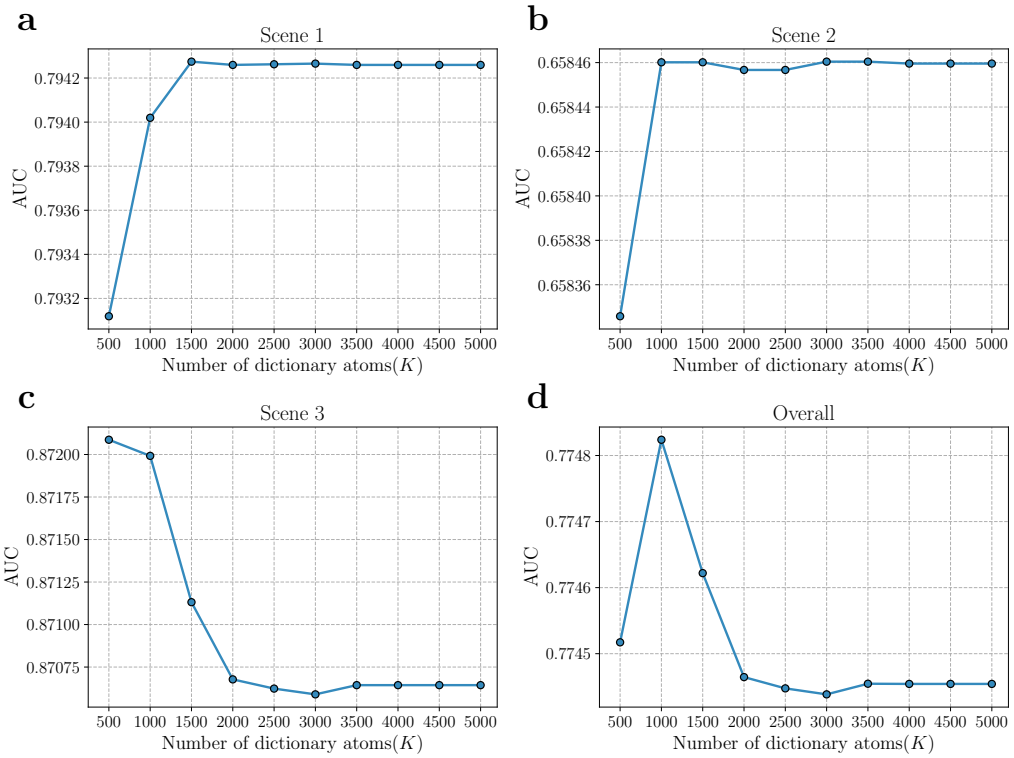

Figure 21: AUC versus number of dictionary atoms (K) on UMN scene 1 **(a)**, scene 2 **(b)**, scene 3 **(c)**, and overall UMN dataset **(d)**.

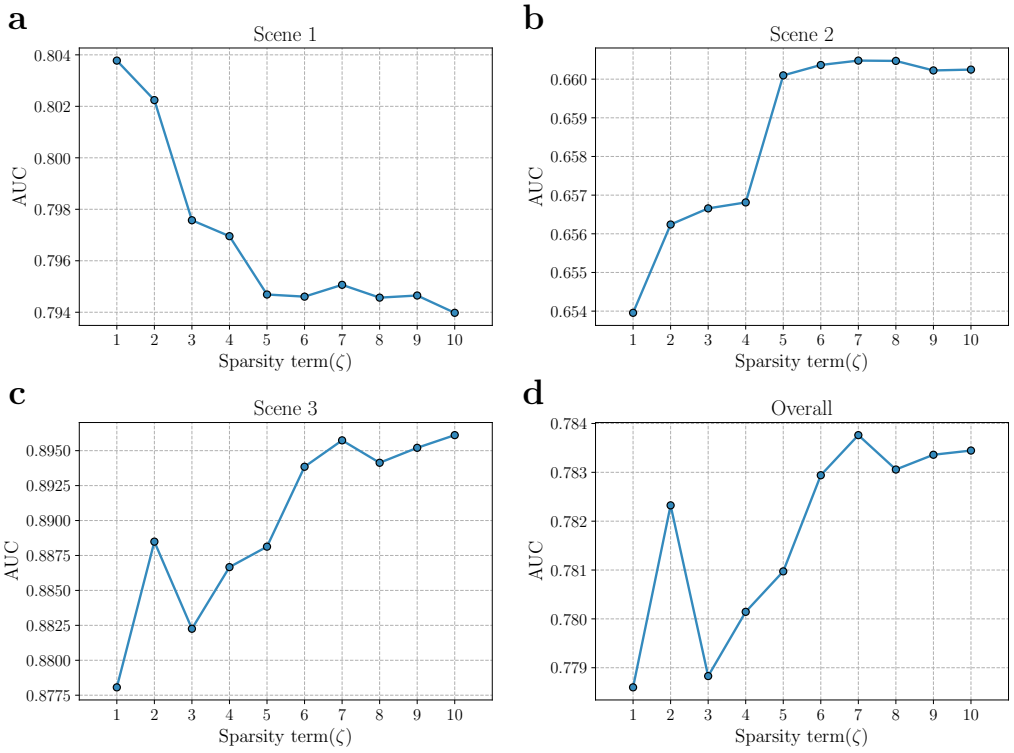

Figure 22: AUC versus sparsity term (ζ) on UMN scene 1 **(a)**, scene 2 **(b)**, scene 3 **(c)**, and overall UMN dataset **(d)**.

Cluster size and cluster removal percentage. Our framework's first stage is mainly controlled by cluster size and cluster removal percentage hyper-parameters. The search space was limited to the cartesian-product of cluster size $= \{10, 20, ..., 100\}$ and cluster removal percentage $= \{0, 20, 40\}$. Cluster removal equals 0% means that all clusters were kept during training. We show the change of AUC with respect to cluster size and cluster removal percentage in Figure 23. It can be seen that removing smaller clusters, as opposed to not removing them at all, always improve the AUC. However, no significant difference for cluster removal percentages 20% and 40% was evident. The greater cluster removal percentage performed better, with some exceptions, *e.g.* at $k = 60$ on UMN scene 1 and 3. On the other hand, we observe an overall decrease in AUC for each scene when the number of clusters increases. The reason may be the small amounts of training samples during the training.

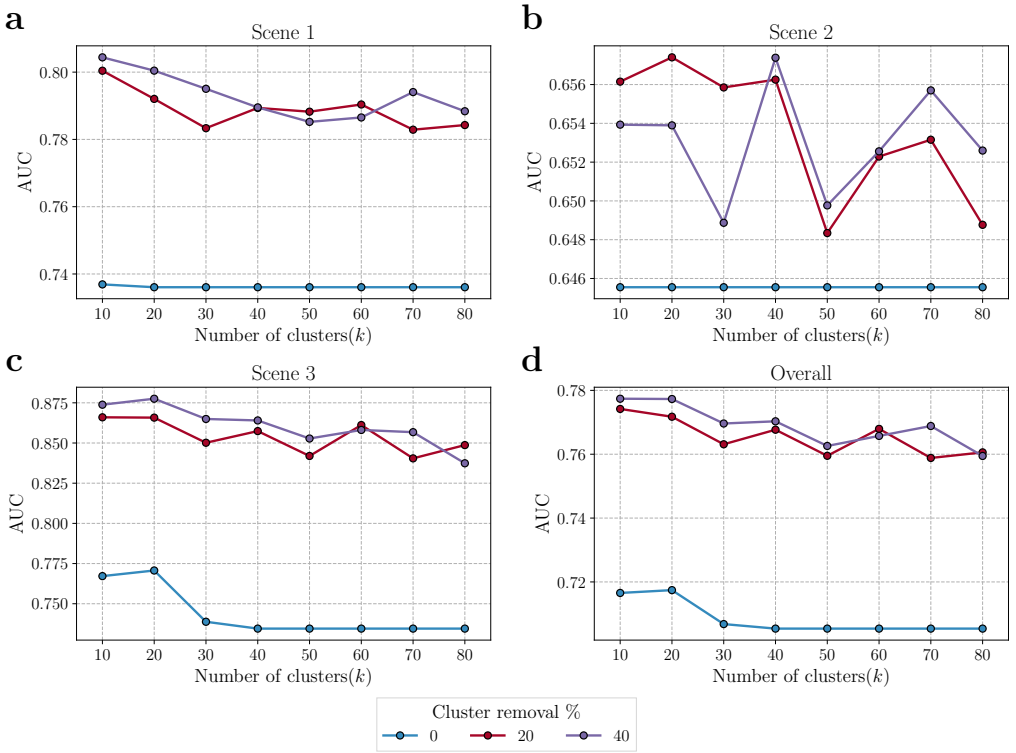

Figure 23: AUC versus cluster size and cluster removal percentage on UMN scene 1 **(a)**, scene 2 **(b)**, scene 3 **(c)**, and overall UMN dataset **(d)**.

Motion response threshold. To determine motion features from the static ones, we apply a certain threshold after the feature extraction. Setting a large threshold eliminates most of the samples while setting a small threshold allows more samples

to be included in the training, but we add less informative features as a trade-off. The search space for finding the optimum motion response threshold was set to $\{1.0, 1.5, ..., 8.0\}$. Figure 24 demonstrates how this hyper-parameter affects AUC on the UMN dataset. Mostly, there is an increase in AUC as the threshold increases. This confirms the fact that allowing more samples, but correspondingly introducing more noisy features, would indeed harm AUC. However, there has been a slight decline in AUC for the threshold greater than 5.5 in Figure 24c. For a feature size $10 \times 10 \times 5$, the optimal threshold varies for each scene (scene 1: 7.5, scene 2: 6.0, scene 3: 5.5). The maximum difference in AUC approximately equals to 0.2, which can be observed in Figure 24a.

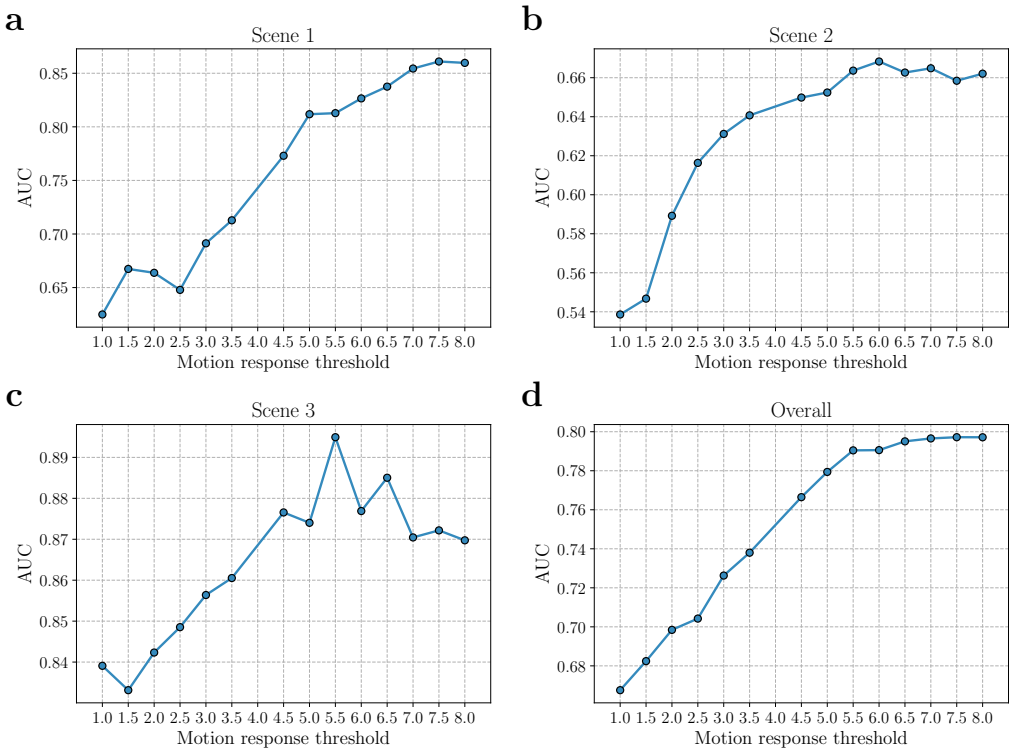

Figure 24: AUC versus motion response threshold on UMN scene 1 **(a)**, scene 2 **(b)**, scene 3 **(c)**, and overall UMN dataset **(d)**.

Post-processing filters. We show the change of AUC for the post-processing filters, *i.e.* three-dimensional uniform and one-dimensional Gaussian filter. The search space was limited to the cartesian-product of a three-dimensional uniform filter size $= \{3, 5, 7, 9, 11\}$ and standard deviation for one-dimensional Gaussian filter $= \{3.0, 5.0, 7.0, 9, 0, 11.0\}$. According to Figure 25, no significant effect of the Gaussian filter in AUC was observed. The reason may be the uniform filter is

already able to smooth the sudden changes good enough, and therefore the Gaussian filter becomes excessive. From the figure, it can be seen that the optimum three-dimensional uniform filter size equals to 3.0, and the bigger filter size always affects the performance poorly.

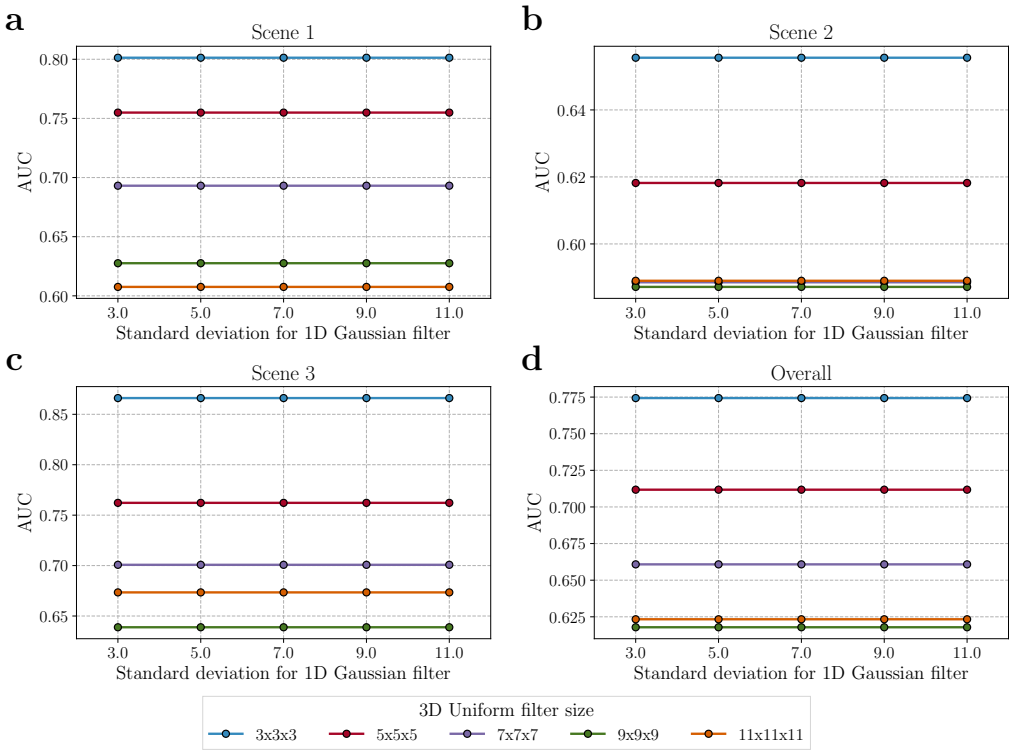

Figure 25: AUC versus 3D uniform filter size and standard deviation for 1D Gaussian filter on UMN scene 1 **(a)**, scene 2 **(b)**, scene 3 **(c)**, and overall UMN dataset **(d)**.

Image size, patch size, and motion response threshold. Depending on the image resolution and patch sizes, local or global abnormal events may be more distinct than the normal events. For example, a local abnormal activity, such as a person pickpocketing may be more noticeable in a scene using a smaller patch size. Therefore, we explored the change of AUC with respect to image size, patch size, and motion response threshold. Repeating the exhaustive search for the optimum threshold was necessary because a motion response threshold may not be suitable for different feature sizes. The search space was set to the cartesian-product of image size $= \{80 \times 60, 160 \times 120, 240 \times 180, 320 \times 240\}$, patch size $= \{5 \times 5, 10 \times 10, 20 \times 20\}$, and motion response threshold $= \{3.0, 6.0, 9.0, 12.0, 15.0\}$. At the end of 60 experiments, exciting results were achieved. First, the superior performance of patch size $= 5 \times 5$ with the threshold $= 3.0$ can be noticed in every setting of the experiment, as shown in Figure 26. In general, patch size $= 20 \times 20$ was the worst

performer compared to the smaller patches; still, the AUC scores were comparable only on the smaller image resolution, especially when image size $= 80 \times 60$. On the other hand, no significant advantage of particular image size under the right motion response threshold setting was seen. Finally, it can be seen that previous results so far were outperformed, with AUC$= 0.832$ (Figure 26o) in the UMN dataset using the following hyper-parameters: image size$= 240 \times 180$, patch size $= 5 \times 5$, and motion response threshold$= 3.0$. In conclusion, a higher resolution image and smaller patch size tend to work well together.

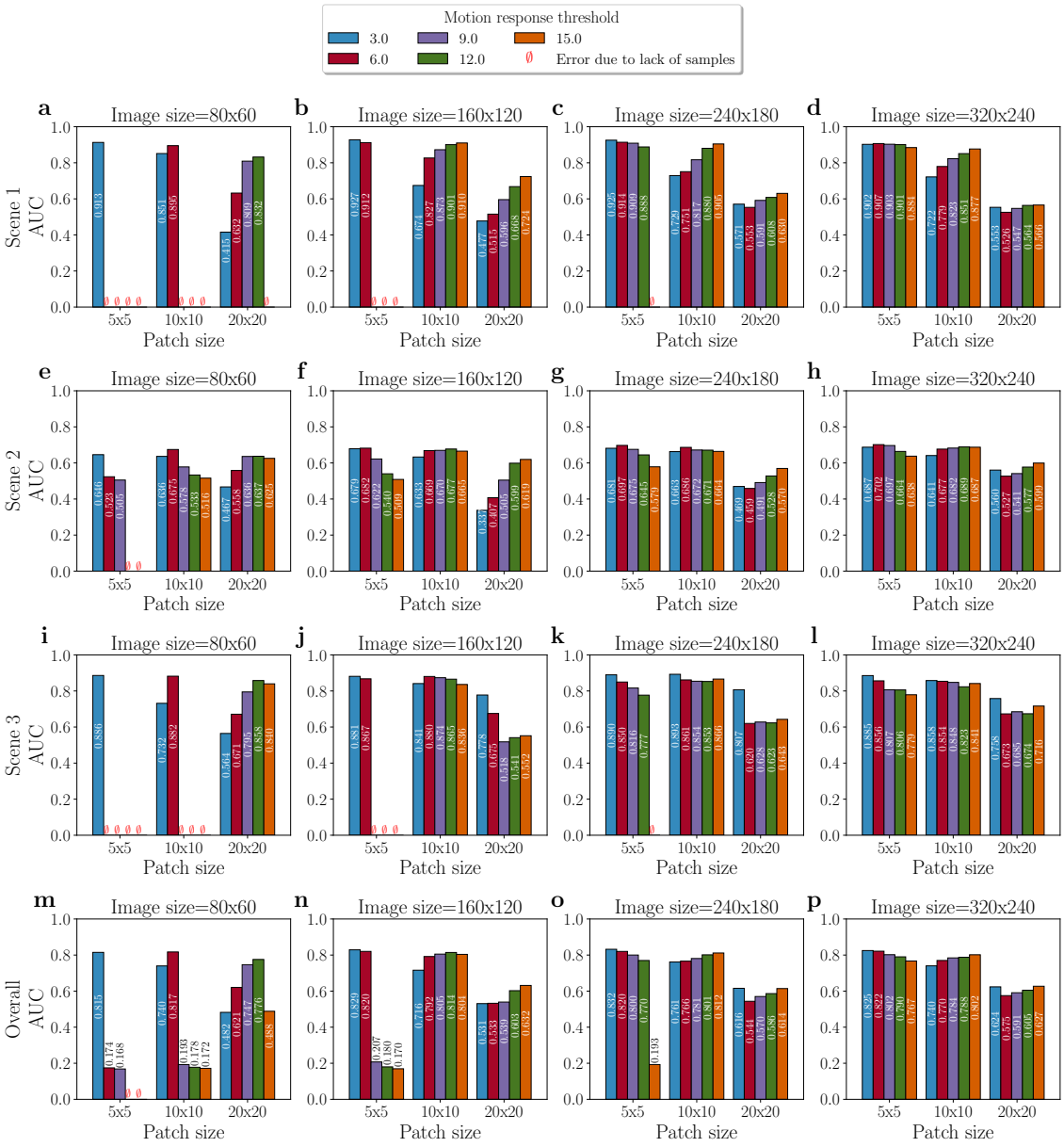

Figure 26: AUC versus image size, patch size, and motion response threshold on UMN scene 1 **(a—d)**, scene 2 **(e—h)**, scene 3 **(i—l)**, and overall UMN dataset **(m—p)**. The runtime errors due to lack of samples as a result of a large motion response threshold are indicated as ∅ on the plots.

5.4 Comparative analysis

We hypothesise that the two-stage sparse representation based approach can contribute to a more robust abnormal event detection. We empirically compared our approach against different methods to reveal the hypothesis under consideration. The competitors, highlighted in bold below, were selected based on the following questions:

- How does our method (**2STG-AKSVD**) perform compared to its base classifier? That is, the approximate K-SVD algorithm (**AKSVD**).
- Do we get comparable result when we use different unsupervised learning algorithms, *e.g.* one-class support vector machine (**OCSVM**) [SPST+01]?
- Does the two-stage approach have a positive effect on abnormal event detection? For example, how does the two-stage one-class support vector machine perform (**2STG-OCSVM**)?

In Section 5.3, the best result was achieved by setting image size= 240×180, patch size = 5×5, and motion response threshold= 3.0. This experiment took 1 hour and 20 minutes to train. We updated the hyper-parameters for our framework according to the best result. No further hyper-parameter search was performed for our framework in this section.

General hyper-parameter selection scheme for the rest of the competitors was the following. All of the methods used the same hyper-parameters in the feature extraction step. Furthermore, the "two-stage" approaches (*i.e.*, 2STG-AKSVD and 2STG-OCSVM) used the same hyper-parameters in the first stage of training (number of clusters, cluster removal percentage *etc.*). Hyper-parameters for the individual unsupervised learning algorithms (*i.e.*, AKSVD and OCSVM) were optimised according to the grid search method, and similarly for the 2STG-OCSVM method as well. We give details of the hyper-parameters of the competitors in the following way.

For AKSVD algorithm, we performed grid search only for two hyper-parameters, and the best-performing ones were: number of dictionary atoms (K) = 5000 and sparsity term (ζ) = 1 non-zero coefficient. Because there was a single dictionary that needed to be trained, the maximum iteration for the dictionary convergence was set to 5000 steps. Each search step took approximately 20 hours to train. The rest of the hyper-parameters were set according to Section 5.1.

The implementation from PyOD Python library was used for the OCSVM algorithm [ZNL19]. In fact, PyOD library implementation is based on *libsvm* [CL11]. A non-linear, radial basis function (RBF) kernel was applied, γ and ν hyper-parameters [SPST+01] were set both as 0.1 according to the grid search result. A search step took on average 25 minutes to train, which is very quick if one compares to previous methods. For other parameters default values were utilised from the library [ZNL19].

Lastly, the hyper-parameters of the 2STG-OCSVM method were set as follows. For the first stage part, the hyper-parameters were set as same as the 2STG-AKSVD method, such as the number of clusters was set to 20, cluster removal percentage was set to 40%, and so on. Next, for each cluster OCSVM was trained with the RBF kernel, γ and ν hyper-parameters [SPST+01] were exhaustively searched. According to the grid search result, the best hyper-parameters were found as $\gamma = 0.002$ and $\nu = 0.5$. On average, a search step took 12 minutes to train, which was even faster than OCSVM. Likewise, for other parameters related to OCSVM, default values were set according to the PyOD library [ZNL19].

A comparison of the competitors in terms of AUC, classification accuracy, F_1 and FPS are presented in Table 6. Moreover, we show the ROC curves in Figure 27. Overall, the performance of AKSVD, 2STG-AKSVD, 2STG-OCSVM were similar. Despite the slight differences, it can be seen from the table that 2STG-OCSVM performed best in AUC= 0.840, and 2STG-AKSVD performed best in classification accuracy= 0.852 and F_1 score= 0.592 on the UMN dataset. According to Figure 27b, all of the methods had a similarly disappointing performance on UMN scene 2, the best performer was OCSVM with AUC=0.692. That is only a 0.192 difference than a random guess classifier. This indicates that changes in other than a black-box classifier is needed to improve the performance on UMN scene 2. Next, AKSVD was the fastest method with 235 FPS on the overall UMN dataset. The two-stage approaches in comparison to their base methods always ran slower. This is not surprising considering that the two-stage approaches perform predictions for each cluster and select the result with a minimum error. With this result, also observe the improvements in classification accuracy and F_1 score of our method over the initial experiment result from Section 5.2.

According to the results on the UMN dataset, using the two-stage approach is ineffective when combined with a sparse representation based method. We observed using AKSVD without the two-stage approach in general not only performed better or equivalent on two out of three scenes, but it was also faster. However, this was

not true in the case of OCSVM and its two-stage version. 2STG-OVSVM was better than OCSVM almost in every classification metric, with an exception only in AUC on UMN scene 2 (Table 6). According to the results, one can conclude that the two-stage approach works well with OCSVM. This finding was in line with the effect of learning explored by Ionescu *et al.* [ISPA18] in "Narrowed Normality Clusters".

UMN dataset	**Metrics**	**Methods**			
		AKSVD[1]	2STG-AKSVD[2]	OCSVM[3]	2STG-OCSVM[4]
Scene 1	AUC	**0.927**	0.925	0.880	0.915
	Accuracy	**0.902**	0.898	0.817	0.852
	F_1	**0.785**	0.773	0.640	0.701
	FPS	**280**	133	133	49
Scene 2	AUC	0.691	0.681	**0.692**	0.690
	Accuracy	**0.770**	0.764	0.751	0.768
	F_1	**0.554**	0.547	0.544	0.549
	FPS	**333**	255	302	155
Scene 3	AUC	0.899	0.890	0.863	**0.914**
	Accuracy	0.879	0.895	0.771	**0.898**
	F_1	0.420	**0.456**	0.272	0.445
	FPS	90	**95**	36	12
Overall	AUC	0.839	0.832	0.812	**0.840**
	Accuracy	0.850	**0.852**	0.780	0.839
	F_1	0.586	**0.592**	0.485	0.565
	FPS	**235**	161	157	72

[1] **AKSVD:** approximate K-SVD. [2] **2STG-AKSVD (proposed):** two-stage approximate K-SVD.
[3] **OCSVM:** one-class support vector machine. [4] **2STG-OCSVM:** two-stage one class-support vector machine.

Table 6: Comparison of AUC, classification accuracy, F_1 score, and FPS on the UMN dataset. The best method under a performance metric is highlighted in bold within the table row. Abbreviations of the competitors are listed at the bottom of the table.

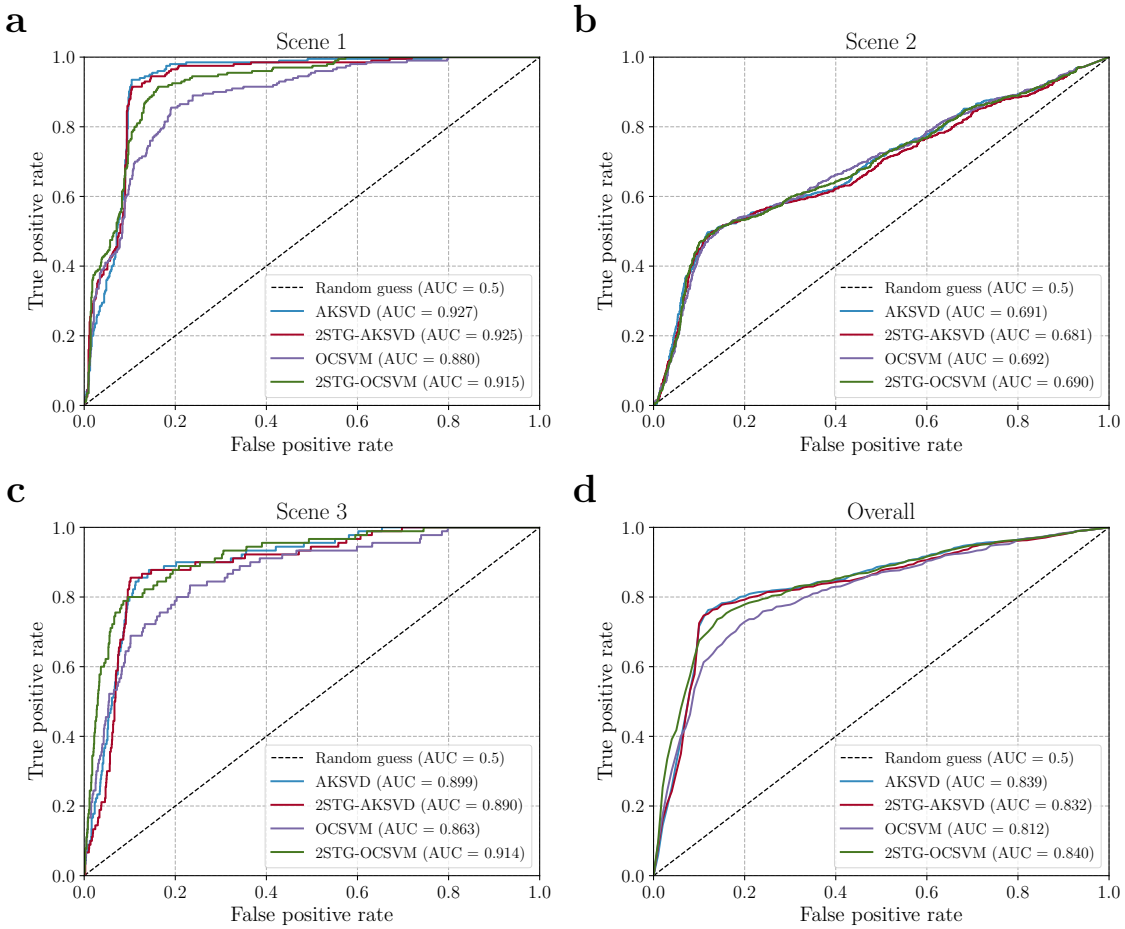

Figure 27: Frame-level comparison of the ROC curves of the competitors on UMN scene 1 **(a)**, scene 2 **(b)**, scene 3 **(c)**, and overall UMN dataset **(d)**. The blue line indicates the ROC curve of AKSVD, the red line 2STG-AKSVD, the purple line OCSVM, and the green line 2STG-OCSVM. The ROC curve of a random guess is shown as dashed black lines. Each AUC of a ROC curve is given in the parenthesis in the figure.

6 Discussion

The main objective of the current study was to implement an abnormal event detection system to automate time-consuming video surveillance task. In an attempt to improve the abnormal event detection performance, we proposed a two-stage approach based on the k-means clustering in the first stage, and sparse representation based methods, such as the batch-OMP and the AKSVD algorithms, in the second stage. Results suggest that our framework can successfully understand unusual crowd activities in a scene, and are comparable to previous research. However, other than slight differences in standard measurements, no significant advantage of using the two-stage sparse representation approach over a single large dictionary was observed. In this section, we first revisit the findings of our study, discuss its limitations and suggest mitigation strategies, where appropriate. Then, we underline unanswered questions for other potential improvements in future research. Lastly, we discuss the usefulness of the study.

Revisiting findings

Contrary to expectations, we did not find a significant difference between AKSVD and its two-stage version. The reason for this is probably that the remaining samples after the feature extraction was too small so features were not effectively separated by k-means clustering. Also, learned dictionaries were similar as a result; the difference between AKSVD and our method was minimal. This contradicts our initial assumption because we expected that as cluster size increase, performance improves (see Figure 23). Nonetheless, further experiments are needed to determine what is the required sample size for the two-stage approach to be useful.

However, in the case of OCSVM and its two-stage version, if we think of cluster centres as something like support vectors, k-means clustering helped in performance improvement regardless of sample size by further narrowing down decision boundaries. Therefore, we confirm the effectiveness of the two-stage OCSVM approach, as suggested by Ionescu *et al.* [ISPA18].

One may claim that the two-stage approach reduces the training time, as suggested by the results. Indeed, that might be certain if we had learned the dictionaries for each cluster in parallel. However, due to hyper-parameter differences, such improvement is not clear to us.

We find the ground truth of the UMN dataset sometimes controversial. According to qualitative results (see Section 5.2), the abnormal activity, *i.e.* a sudden crowd dispersal, had begun earlier than what the ground truth indicates. Because of this, our method generated numerous false positives for each scene by alerting it early. One explanation could be the annotation strategy of the UMN dataset; but, to date, this remains unknown [MOS09]. Biswas and Babu made a similar observation, where their method got 0.74 AUC on the original and 0.95 AUC on the modified ground truth version of the UMN dataset [BB13]. They also experimented on the UCSD dataset [MLBV10], which is considered as a more challenging dataset, and obtained 0.78 AUC. We did not propose such a modification in this study. But we believe that such a modification or a clarification on the annotation strategy is further needed. Were the experiments to be repeated, we would suggest another abnormal crowd activity dataset to be used due to the confusions the UMN dataset ground truth.

One positive outcome of this project was the runtime performance. As mentioned in the literature review section, many studies either do not performed performance analysis or their methods were not suitable for testing in real-time. Due to this prior work, the runtime performance of the selected methods was worrying. However, the results on the UMN dataset show that our framework is suitable for a real-time application. Indeed, the primary reason is the use of the parallel and C ++ implementations of the batch-OMP and AKSVD algorithms. Initially, we implemented in pure Python and utilised efficient packages, such as SciPy [JOP01], NumPy [WCV11], but still, the C++ implementation of the algorithms was approximately ten times faster. From a theoretical aspect, the use of a greedy algorithm, compared to ℓ_1-norm solvers, on sparse code generation is likely another factor. If Ren *et al.*'s findings were accurate; greedy algorithms require less computation time, and the OMP algorithm is approximately 180 times faster than the LASSO algorithm [RPO+16]. Another reason is the use of simple and fast, spatio-temporal gradient feature in event representation, which consists of a few matrix operations. Due to these design choices made, our project has reached real-time video processing speed.

Limitations of the study

This study had several limitations that should be mentioned. Firstly, we describe the shortcomings of the method. Then, we talk about problems which are introduced by the experimental setup.

Shortcomings of the method

Despite the advantages, there are also various limitations due to the feature extraction method used by this study. First, long-term temporal relationships in a video were not effectively used. In experiments, our method used only five consecutive frames to represent events in a spatio-temporal volume. It represents a short time. While sudden changes in a short time may mean an anomaly, such as a panic situation, in a longer duration we may also understand more complex behaviours. Cong et al. used 16 frames in their method for UCSD Peds 1 dataset, whereas they used five frames for the Subway dataset [CYL13]. This tells us there is no one-size-fits-all solution for a temporal window size hyper-parameter. We did not analyse for the hyper-parameter due to a small number of samples. Finding the right threshold to generate useful features, in general, was challenging in the study. Therefore, we confirm the finding by Mabrouk and Zagrouba, who indicated the feature's sensitivity to hyper-parameters [MZ18].

Second, we disregarded colour data in a video because we converted all images to grayscale. Colour may serve as a useful appearance feature to understand certain anomalies, for example, fire detection from a scene. An intuitive idea would be to build feature vectors around a colour model.

The last limitation is related to the feature's sensitivity to illumination changes, *e.g.* we saw a false positive prediction due to this in Section 5.2. Illumination is a common problem in computer vision [PW12, SRB12]. Solutions typically involve a colour space transformation or a normalisation step to make features illumination invariant. All these limitations necessitate seeking for an alternative feature extraction technique.

Let us move now to the limitations due to the learning algorithms. Our framework requires a "re-training" process to adapt to the changes in a scene, such as daylight or environmental changes, because of the offline learning approach. We could address this issue by a global and local dictionary update strategy similar to Lv *et al.*'s approach [LLXZ18]. In a global update, a dictionary is completely replaced for new data. In a local update, a pre-trained dictionary is further iterated in the dictionary update step for new data. A threshold value decides whether if a global or local dictionary update should be carried out.

We noticed that the training time of the AKSVD algorithm, even for a small size dataset, was higher than the OCSVM. We note generally sparse representation based

methods require high training time, *e.g.* [LSJ13, YLS17]. Lu *et al.* addressed this issue by proposing the online version sparse combination learning algorithm [LSWJ18]. In the same vein, we could utilise the online dictionary learning algorithm by Mairal *et al.* to improve the training time [MBPS09].

Lastly, we used the same hyper-parameters while learning a dictionary for each cluster. This reduced the complexity of selecting hyper-parameters in the dictionary learning (*i.e.*, K and ζ). However, we likely introduced non-optimal hyper-parameters for the dictionaries. Fortunately, its implication on the detection performance should not be so crucial according to the results.

Problems with the experimental setup.

We are aware that our experimental setup may have some limitations. We followed the same experimental setup in training, where N first frames were used for training and rest for testing as suggested by previous research. Although this is widely accepted, one problem is that model tuning without cross-validation yields overfit. Moreover, we used the exhaustive grid search method on pre-defined splits because of the lack of cross-validation. Another is the lack of a mechanism to attenuate the class imbalance problem.

Applying cross-validation is a typical approach to solve overfitting a model. However, we find it particularly difficult to apply, *e.g.* k-fold cross-validation without ignoring the temporal elements in the problem. The question, does one apply cross-validation on a frame or feature level, and how to ensure the proper split strategy? From the literature review, we identified only two studies perform cross-validation on their methods, but the details are not given [HSS18, CXYC19]. Liu *et al.* suggest a leave-one-out cross-validation (LOOCV) strategy in action recognition task [LLS09]. Similarly, for anomaly detection, LOOCV could be applied at video-level, *e.g.* for each scene in the UMN dataset. This way, the hyper-parameters in our method could be chosen without risk for overfitting.

Due to a lack of cross-validation, a brute-force approach was performed to alleviate the technical overhead, and as a result, computational resources were wasted. Given that LOOCV is applied, we could utilise the *Random Search* method, which is known to yield faster and better hyper-parameters than human experts [BB12]. We note in our literature review typically no details of hyper-parameter search methods are indicated by the studies.

The UMN dataset was imbalanced so that more frames were identified as normal rather than abnormal. The effect on our study is that skewed data may bias some performance metrics. The class imbalance is a common problem in anomaly detection. Interestingly, we note that previous studies have not dealt with the class imbalance problem in much detail. Image data augmentation techniques, such as the horizontal flip of an image using `imgaug` package could be applied to increase the minority class [JWC+20].

Unanswered questions

Several questions remained unanswered in our study. First, several hyper-parameters were left out from the analysis due to the scope; the followings could be investigated more: temporal window size, frame stride, dictionary initialisation scheme, and feature normalisation methods. From a practical point of view, we note that most of the researchers in our review implemented their methods in MATLAB. What is the advantage of MATLAB's image processing toolbox over the tools we use? We used the reconstruction error computed by the batch-OMP algorithm; Lu *et al*. approximated this using the ordinary least squares solution to speed up their method in testing [LSJ13]. What is the runtime advantage, and performance trade-off of using the approximated reconstruction error? Many studies [LSJ13, YLS17, LSWJ18] utilise pyramid representation to improve the detection performance, which means they combine multiple resolutions of an image from a scene. What is the performance gain of using multiple scales over a single scale image in feature extraction?

Usefulness of the study

Despite the limitations, this study is the first step towards enhancing our understanding of a challenging computer vision task, abnormal event detection. Results have been encouraging, and it motivates us to create improvements. Due to its unsupervised learning nature, our framework can be adapted to different contexts without additional annotation effort and using only normal events from videos. It could be a part of an ensemble learning architecture to improve abnormal detection performance. Moreover, the methods used for understanding anomalies may be applied to the action recognition task as well, by training binary classifiers for each action.

7 Conclusion

This study proposed a two-stage sparse representation based abnormal crowd event detection framework. The main contributions of this thesis can be summarised in three aspects.

First, we carried out a literature review of 18 studies, in which we focused solely on methods related to sparse representation. We identified that most of the studies either could not operate in real-time or did not include any runtime analysis. Motivated by the studies in our review, we selected the spatio-temporal gradient feature for event representation. If a goal is to understand the motion in short-term changes, *e.g.* in five consecutive frames, in a simple and fast manner, then we also recommend the spatio-temporal gradient feature to be used.

Second, the proposed abnormal event detection framework performed 0.832 AUC, 0.852 classification accuracy, and 0.592 F_1 score on the UMN dataset [MOS09]. Moreover, we were able to meet the real-time video processing requirement with 161 FPS. We presented that when parallel and efficient implementations were employed, sparse representation based methods are also able to run fast.

Lastly, we analysed the effects of hyper-parameters and empirically compared our framework against its base classifier (*i.e.*, AKSVD), one-class support vector machine (OCSVM), and two-stage version of OCSVM. No significant effect of the two-stage approach was found on the performance when combined with AKSVD, but performance improvement was evident with OCSVM. We speculate that this might be due to the small sample size. Nevertheless, we recommend using a single large dictionary, which works slightly better or equivalent, and much faster on the UMN dataset.

The most important limitation lies in the fact that our approach has not fully exploited the temporal and colour data of a video. Thus, we plan to improve the performance by combining with deep learned features that can extract long-term dependencies. One approach would be to use the pre-trained convolutional neural network (CNN) for action recognition task proposed by Tran *et al.* [TWT+18]. We can use the pre-trained network to extract appearance and motion features from a video and adapt for abnormal detection task by employing transfer learning. We also aim to evaluate our approach on other standard datasets [ARSR08, MLBV10, LSJ13].

References

[AEB06] Aharon, M., Elad, M. and Bruckstein, A., K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54,11(2006), pages 4311–4322.

[ARSR08] Adam, A., Rivlin, E., Shimshoni, I. and Reinitz, D., Robust real-time unusual event detection using multiple fixed-location monitors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30,3(2008), pages 555–560.

[AV07] Arthur, D. and Vassilvitskii, S., k-means++: The advantages of careful seeding. *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2007, pages 1027–1035.

[BB12] Bergstra, J. and Bengio, Y., Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13,Feb(2012), pages 281–305.

[BB13] Biswas, S. and Babu, R. V., Real time anomaly detection in h. 264 compressed videos. *2013 Fourth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)*. IEEE, 2013, pages 1–4.

[BD09] Blumensath, T. and Davies, M. E., Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27,3(2009), pages 265–274.

[Bra00] Bradski, G., The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

[BT09] Beck, A. and Teboulle, M., A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2,1(2009), pages 183–202.

[CL11] Chang, C.-C. and Lin, C.-J., LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2,3(2011), pages 1–27.

[CWZ+17] Chen, Z. ., Wu, C. ., Zhang, Y. ., Huang, Z., Jiang, J. ., Lyu, N. . and Ran, B., Vehicle behavior learning via sparse reconstruction with 2 - p minimization and trajectory similarity. *IEEE Transactions on Intelligent Transportation Systems*, 18,2(2017), pages 236–247.

[CXYC19] Chu, W., Xue, H., Yao, C. and Cai, D., Sparse coding guided spatiotemporal feature learning for abnormal event detection in large videos. *IEEE Transactions on Multimedia*, 21,1(2019), pages 246–255.

[CYL13] Cong, Y., Yuan, J. and Liu, J., Abnormal event detection in crowded scenes using sparse representation. *Pattern Recognition*, 46,7(2013), pages 1851–1864.

[DBR16] Dutta, J. K., Banerjee, B. and Reddy, C. K., Rods: Rarity based outlier detection in a sparse coding framework. *IEEE Transactions on Knowledge and Data Engineering*, 28,2(2016), pages 483–495.

[DGBH16] Del Giorno, A., Bagnell, J. A. and Hebert, M., A discriminative framework for anomaly detection in large videos. *European Conference on Computer Vision*. Springer, 2016, pages 334–349.

[Doy16] Doyle, S., Video analytics adoption: Key considerations for the end-user. Technical Report, Publications Office of the European Union, 2016.

[EAH99] Engan, K., Aase, S. O. and Husoy, J. H., Method of optimal directions for frame design. *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings.*, volume 5. IEEE, 1999, pages 2443–2446.

[EHJ+04] Efron, B., Hastie, T., Johnstone, I., Tibshirani, R. et al., Least angle regression. *The Annals of Statistics*, 32,2(2004), pages 407–499.

[GDL19] Geng, Y., Du, J. and Liang, M., Abnormal event detection in tourism video based on salient spatio-temporal features and sparse combination learning. *World Wide Web*, 22,2(2019), pages 689–715.

[HFWW13] Han, S., Fu, R., Wang, S. and Wu, X., Online adaptive dictionary learning and weighted sparse coding for abnormality detection. *2013 IEEE International Conference on Image Processing, ICIP 2013 - Proceedings*, 2013, pages 151–155.

[HSS18] He, C., Shao, J. and Sun, J., An anomaly-introduced learning method for abnormal event detection. *Multimedia Tools and Applications*, 77,22(2018), pages 29573–29588.

[ISPA18] Ionescu, R. T., Smeureanu, S., Popescu, M. and Alexe, B., Detecting abnormal events in video using narrowed motion clusters. *arXiv preprint arXiv:1801.05030*.

[JOP01] Jones, E., Oliphant, T. and Peterson, P., Scipy: Open source scientific tools for python.

[JWC+20] Jung, A. B., Wada, K., Crall, J., Tanaka, S., Graving, J., Reinders, C., Yadav, S., Banerjee, J., Vecsei, G., Kraft, A., Rui, Z., Borovec, J., Vallentin, C., Zhydenko, S., Pfeiffer, K., Cook, B., Fernández, I., De Rainville, F.-M., Weng, C.-H., Ayala-Acevedo, A., Meudec, R., Laporte, M. et al., imgaug, `https://github.com/aleju/imgaug`, 2020. Online; accessed 01-Feb-2020.

[KN09] Kratz, L. and Nishino, K., Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models. *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pages 1446–1453.

[LCW+15] Li, T., Chang, H., Wang, M., Ni, B., Hong, R. and Yan, S., Crowded scene analysis: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 25,3(2015), pages 367–386.

[LLG17] Luo, W., Liu, W. and Gao, S., A revisit of sparse coding based anomaly detection in stacked RNN framework. *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pages 341–349.

[Llo82] Lloyd, S., Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28,2(1982), pages 129–137.

[LLS09] Liu, J., Luo, J. and Shah, M., Recognizing realistic actions from videos "in the wild". *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pages 1996–2003.

[LLXZ18] Lv, P., Liu, S., Xu, M. and Zhou, B., Abnormal event detection and location for dense crowds using repulsive forces and sparse reconstruction. *arXiv preprint arXiv:1808.06749*.

[LSJ13] Lu, C., Shi, J. and Jia, J., Abnormal event detection at 150 fps in MATLAB. *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pages 2720–2727.

[LSWJ18] Lu, C., Shi, J., Wang, W. and Jia, J., Fast abnormal event detection. *International Journal of Computer Vision*, pages 1–19.

[MBP+14] Mairal, J., Bach, F., Ponce, J. et al., Sparse modeling for image and vision processing. *Foundations and Trends® in Computer Graphics and Vision*, 8,2-3(2014), pages 85–283.

[MBPS09] Mairal, J., Bach, F., Ponce, J. and Sapiro, G., Online dictionary learning for sparse coding. *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pages 689–696.

[MH17] Masoudirad, S. M. and Hadadnia, J., Anomaly detection in video using two-part sparse dictionary in 170 fps. *2017 3rd International Conference on Pattern Recognition and Image Analysis (IPRIA)*. IEEE, 2017, pages 133–139.

[MLBV10] Mahadevan, V., LI, W.-X., Bhalodia, V. and Vasconcelos, N., Anomaly detection in crowded scenes. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pages 1975–1981.

[MOS09] Mehran, R., Oyama, A. and Shah, M., Abnormal crowd behavior detection using social force model. *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pages 935–942.

[MZ93] Mallat, S. G. and Zhang, Z., Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41,12(1993), pages 3397–3415.

[MZ18] Mabrouk, A. B. and Zagrouba, E., Abnormal behavior recognition for intelligent video surveillance systems: A review. *Expert Systems with Applications*, 91, pages 480–491.

[OF96] Olshausen, B. A. and Field, D. J., Natural image statistics and efficient coding. *Network: Computation in Neural Systems*, 7,2(1996), pages 333–339.

[OF97] Olshausen, B. A. and Field, D. J., Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37,23(1997), pages 3311–3325.

[PRK93] Pati, Y. C., Rezaiifar, R. and Krishnaprasad, P. S., Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*. IEEE, 1993, pages 40–44.

[PVG+11] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. et al., Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12,Oct(2011), pages 2825–2830.

[PW12] Popoola, O. P. and Wang, K., Video-based abnormal human behavior recognition—a review. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42,6(2012), pages 865–878.

[RPO+16] Ren, H., Pan, H., Olsen, S. I., Jensen, M. B. and Moeslund, T. B., An in-depth study of sparse codes on abnormality detection. *2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2016, pages 66–72.

[RZE08] Rubinstein, R., Zibulevsky, M. and Elad, M., Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit. Technical Report, Computer Science Department, Technion, 2008.

[SFH16] Sabokrou, M., Fathy, M. and Hoseini, M., Video anomaly detection and localisation based on the sparsity and reconstruction error of auto-encoder. *Electronics Letters*, 52,13(2016), pages 1122–1124.

[SPST+01] Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J. and Williamson, R. C., Estimating the support of a high-dimensional distribution. *Neural Computation*, 13,7(2001), pages 1443–1471.

[SRB12] Sodemann, A. A., Ross, M. P. and Borghetti, B. J., A review of anomaly detection in automated surveillance. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42,6(2012), pages 1257–1272.

[SWXS18] Sun, J., Wang, X., Xiong, N. and Shao, J., Learning sparse representation with variational auto-encoder for anomaly detection. *IEEE Access*, 6, pages 33353–33361.

[Tib96] Tibshirani, R., Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58,1(1996), pages 267–288.

[TWT+18] Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y. and Paluri, M., A closer look at spatiotemporal convolutions for action recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pages 6450–6459.

[WCV11] Walt, S. v. d., Colbert, S. C. and Varoquaux, G., The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13,2(2011), pages 22–30.

[XDS+11] Xu, J., Denman, S., Sridharan, S., Fookes, C. and Rana, R., Dynamic texture reconstruction from sparse codes for unusual event detection in crowded scenes. *Proceedings of the 2011 Joint ACM Workshop on Modeling and Representing Events*. ACM, 2011, pages 25–30.

[XLJ16] Xue, J., Lu, Y. and Jiang, H., Abnormal event detection and localization by using sparse coding and reconstruction. *Advances in Multimedia Information Processing - PCM 2016*, volume 9917 LNCS, 2016, pages 306–314.

[YFL18] Yuan, Y., Feng, Y. and Lu, X., Structured dictionary learning for abnormal event detection in crowded scenes. *Pattern Recognition*, 73, pages 99–110.

[YLS17] Yu, B., Liu, Y. and Sun, Q., A content-adaptively sparse reconstruction method for abnormal events detection with low-rank property. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47,4(2017), pages 704–716.

[YLZ+16] Yi, Y., Li, X., Zhao, R., Bi, C., Wang, J. and Sun, H., A constrained sparse representation approach for video anomaly detection. *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, 2016, pages 45–49.

[ZFFX11] Zhao, B., Fei-Fei, L. and Xing, E. P., Online detection of unusual events in videos via dynamic sparse coding. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2011, pages 3313–3320.

[ZNL19] Zhao, Y., Nasrullah, Z. and Li, Z., PyOd: A python toolbox for scalable outlier detection. *arXiv preprint arXiv:1901.01588*.